(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 March 2004 (25.03.2004)

PCT

(10) International Publication Number
WO 2004/025469 A1

(51) International Patent Classification[7]: G06F 11/00

(21) International Application Number:
PCT/US2003/028149

(22) International Filing Date:
9 September 2003 (09.09.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/409,425 10 September 2002 (10.09.2002) US

(71) Applicant *(for all designated States except US)*: **UNION SWITCH & SIGNAL, INC.** [US/US]; 1000 Technology Drive, Pittsburgh, PA 15219-3120 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **BLEVINS, Joseph, S., Sr.** [US/US]; 736B NE Sunnyside School Road, Blue Springs, MO 64014 (US).

(74) Agent: **Houser, Kirk, D.**; Eckert Seamans Cherin & Mellott, LLC, 600 Grant, 44th Floor, Pittsburgh, PA 15219 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

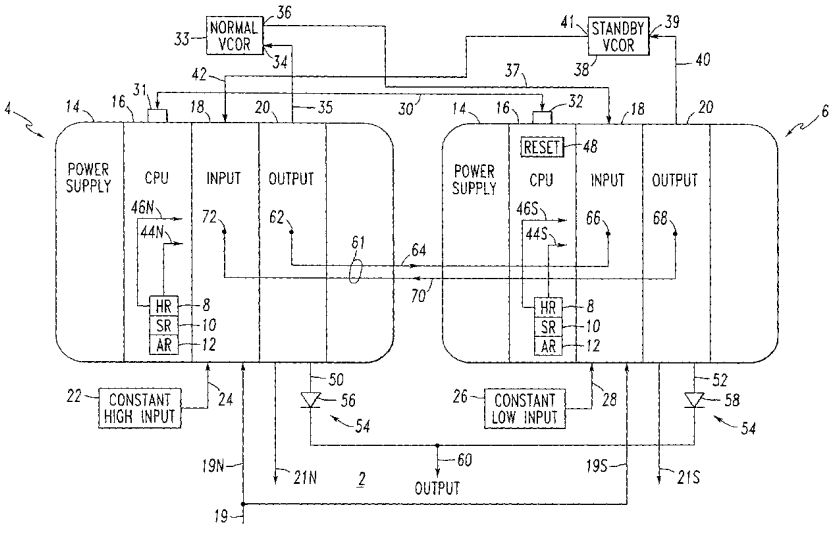
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES,*

[Continued on next page]

(54) Title: HOT STANDBY METHOD AND APPARATUS



(57) Abstract: An apparatus provides hot standby operation with normal and standby processors, each of which includes vital inputs electrically interconnected with the vital inputs of the other processor, vital outputs, and an application routine inputting the vital inputs and outputting the vital outputs. Communication ports communicate with communication ports of the other processor. A health routine provides a health status after communication is established with the other processor. A vital relay includes an input controlled by a vital output and an output to a vital input of the other processor. A synchronization routine provides a synchronization status through the communication ports. The application routine outputs the vital outputs when the synchronization status is set. The standby processor includes a reset routine, which resets the standby processor when the health status of that processor is not provided. A vital "OR" circuit outputs from the vital outputs of the normal and standby processors.

WO 2004/025469 A1

FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

HOT STANDBY METHOD AND APPARATUS

CROSS REFERENCE TO RELATED APPLICATION

This application claims the benefit of U.S. Provisional Application Serial No. 60/409,425, filed September 10, 2002.

COMPUTER PROGRAM LISTING APPENDICES

This application includes three computer program Appendices A, B and C, which are hereby incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to apparatus including normal and standby processors and, more particularly, to such apparatus providing hot standby operation. The invention also relates to a method for providing hot standby operation with normal and standby processors.

Background Information

U.S. Patent No. 5,794,167 discloses a rail transport microprocessor based reliability system for monitoring and controlling actuators as a function of data supplied by sensors. The system includes at least two parallel microprocessors handling the same application. The microprocessors receive pre-encoded data from the sensors and microprocessor output data. A third, comparison microprocessor, known as a voter, employs software to compare the encoded characteristic results of the respective parallel microprocessors.

U.S. Patent No. 4,181,945 discloses a high-reliability vehicle control system including two redundant computer systems. Each of the computer systems consists of two computers, which compare their results and deliver them only if they agree. Which one of the two computer systems processes telegrams received from a control center and compiles telegrams to the control center from messages of on-board units is determined from the control center. At regular intervals, switchover to the other computer system is effected to check whether the latter is functioning correctly or not.

U.S. Patent No. 6,281,606 discloses a plural output electric train control station, which employs a data processor for monitoring and controlling signals generated at a plurality of transformer-driven power output terminals.

U.S. Patent No. 5,751,569 discloses a method of controlling railroad train movement over a layout of railroad track, which is defined geographically using a linear network of geographic control objects. A train control process may be distributed (*e.g.*, not requiring a single central processing unit) and lends itself to localized testing when a failed hardware module is replaced, as only the function performed by that module need be tested.

U.S. Patent No. 5,301,906 discloses an Interlocking Control System (ICS), such as the Microlok® railroad interlocking control system for railroad switching and signaling. A signal to move a switch to its normal position, for example, may be produced in three controllers. Input/output signals regarding entrance and exit locations in a shared territory are transferred between a control console and a terminal block over a data communication link. This information is further transferred in parallel fashion between the controllers and terminal block over respective data communication links. Signals to and from the field are respectively transferred in parallel fashion between the controllers and the terminal block over respective data communication links. Signals output from the controllers are respectively fed via lines to common connection at a node. Interposing diodes are provided to prevent undesired backfeed. A similar terminal connection is employed for outputting a common signal to the three controllers.

There is room for improvement in apparatus and methods for providing hot standby operation with normal and standby processors.

SUMMARY OF THE INVENTION

This need and others are met by the present invention. Many of the past hindrances to develop a hot standby Microlok® have been due to an inability to remain focused on the fundamental reason for a hot standby. A hot standby is for the purpose of having hardware backup not logic backup. Since one Microlok® unit is capable of providing failsafe operation, an additional unit is not for the purpose of making the system more failsafe, it is simply providing a backup system that can be

utilized until a maintainer can be dispatched to repair the hardware of the primary unit.

In accordance with one aspect of the invention, an apparatus for providing hot standby operation comprises: a normal processor; a standby processor; each of the normal and standby processors comprising: a plurality of vital inputs, at least some of the vital inputs being electrically interconnected with at least some of the vital inputs of the other one of the standby and normal processors, a plurality of vital outputs, means for communicating with the other one of the standby and normal processors, a health routine providing a health status after communication is established with the other one of the standby and normal processors through the means for communicating, a vital relay including an input controlled by one of the vital outputs and an output to one of the vital inputs of the other one of the standby and normal processors, a synchronization routine providing a synchronization status through the means for communicating with the other one of the standby and normal processors, and an application routine outputting the vital outputs when the synchronization status is set and inputting the vital inputs; the standby processor further comprising a reset routine, which resets the standby processor when the health status of the standby processor is not provided; and means for outputting from some of the vital outputs of the normal processor and from some of the vital outputs of the standby processor.

The normal and standby processors may operate in at least one mode selected from the group comprising: a first mode wherein both of the normal and standby processors output through at least one of the some of the vital outputs of the normal and standby processors, respectively, without restriction; a second mode wherein the normal processor outputs through at least one of the some of the vital outputs of the normal processor without restriction and the standby processor verifies through the means for communicating of the standby processor that the standby processor agrees with the normal processor before outputting through at least one of the some of the vital outputs of the standby processor and, otherwise, the standby processor being reset; and a third mode wherein both of the normal and standby processors verify through the means for communicating of the normal and standby processors, respectively, that the normal and standby processors, respectively, agree

with the standby and normal processors, respectively, before outputting through at least one of the some of the vital outputs of the normal and standby processors, respectively, and, otherwise, the normal and standby processors being reset.

The normal and standby processors may operate in modes wherein the normal processor outputs through at least one of the some of the vital outputs of the normal processor without restriction and the standby processor verifies through the means for communicating of the standby processor that the standby processor agrees with the normal processor before outputting through at least one of the some of the vital outputs of the standby processor and, otherwise, the standby processor being reset.

The means for outputting may include a vital OR circuit having a first input from one of the some of the vital outputs of the normal processor, a second input from one of the some of the vital outputs of the standby processor, and an output adapted to output to a single output device.

The health routine of the normal and standby processors may periodically exchange health information with the health routine of the standby and normal processors, respectively, in order to provide the health status when the one of the vital inputs of the other one of the standby and normal processors is set and the health information is periodically received.

In accordance with another aspect of the invention, a hot standby method comprises: employing a normal processor; employing a standby processor; with each of the normal and standby processors: employing a plurality of vital inputs, electrically interconnecting at least some of the vital inputs with at least some of the vital inputs of the other one of the standby and normal processors, employing a plurality of vital outputs, communicating with the other one of the standby and normal processors, providing a health status after communication is established with the other one of the standby and normal processors, employing a vital relay including an input controlled by one of the vital outputs and an output to one of the vital inputs of the other one of the standby and normal processors, providing a synchronization status associated with the communicating with the other one of the standby and normal processors, and employing an application routine for outputting the vital outputs when the synchronization status is set and inputting the vital inputs; employing with the

standby processor a reset routine, which resets the standby processor when the health status of the standby processor is not provided; and outputting from some of the vital outputs of the normal processor and from some of the vital outputs of the standby processor.

In accordance with another aspect of the invention, a method for providing normal and standby processors comprises: employing a normal processor; employing a standby processor; with each of the normal and standby processors: employing a plurality of vital inputs, electrically interconnecting at least some of the vital inputs with at least some of the vital inputs of the other one of the standby and normal processors, employing a plurality of vital outputs, communicating with the other one of the standby and normal processors, providing a health status after communication is established with the other one of the standby and normal processors, employing a vital relay including an input controlled by one of the vital outputs and an output to one of the vital inputs of the other one of the standby and normal processors, providing a synchronization status associated with the communicating with the other one of the standby and normal processors, and employing an application routine for outputting the vital outputs when the synchronization status is set and inputting the vital inputs; employing with the standby processor a reset routine, which resets the standby processor when the health status of the standby processor is not provided; outputting from some of the vital outputs of the normal processor and from some of the vital outputs of the standby processor; and disabling the some of the vital outputs of the standby processor if the output of the vital relay of the normal processor is set.

BRIEF DESCRIPTION OF THE DRAWINGS

A full understanding of the invention can be gained from the following description of the preferred embodiments when read in conjunction with the accompanying drawings in which:

Figure 1 is a block diagram of an interlocking control system including a normal processor and a standby processor in accordance with the present invention.

Figure 2 is a schematic diagram of an OR circuit including a pair of diodes for the normal and standby processors of Figure 1, in which both units actively produce outputs at all times.

Figure 3 is a schematic diagram of an OR circuit including a pair of diode arrays for the normal and standby processors of Figure 1, in which both units actively produce outputs at all times.

Figure 4 is a flow diagram showing the signal interconnections between the normal and standby processors in accordance with an embodiment of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The method and apparatus disclosed herein is applied to an Interlocking Control System (ICS), such as the Microlok® railroad interlocking control system for railroad switching and signaling, as described in U.S. Patent No. 5,301,906, which is hereby incorporated herein by reference. Although Microlok® units are disclosed, the invention is applicable to other ICS signal equipment, railway control circuitry, railway signaling, and railway logic devices, such as, for example, a Microlok® II Wayside Control System marketed by Union Switch & Signal, Inc. of Pittsburgh, Pennsylvania.

Example 1

Referring to Figure 1, an apparatus, such as an Interlocking Control System (ICS) 2, provides hot standby operation. The ICS 2 includes a normal processor unit 4 and a standby processor unit 6. In accordance with the present invention, the units 4,6 of the ICS 2 include a health routine (HR) 8, a synchronization routine (SR) 10, and an application routine (AR) 12. The unit 6 also includes a reset routine 48.

Each of the processor units 4,6 includes a power supply 14, a central processing unit (CPU) 16, one or more vital input boards 18 (only one is shown with each of the units 4,6) inputting a plurality of vital inputs 19, and one or more vital output boards 20 (only one is shown with each of the units 4,6) outputting a plurality of vital outputs 21N,21S, respectively. Preferably, all of the vital inputs 19N of the normal vital input board 18 are electrically interconnected with the vital inputs 19S of the standby vital input board 18. The normal state of the normal unit 4 is defined by a constant high input 22, which is applied to one of the normal vital inputs 24. The standby state of the standby unit 6 is defined by a constant low input 26, which is applied to one of the standby vital inputs 28.

A suitable communication channel 30 is provided for communicating between each of the normal and standby CPUs 16, which CPUs respectively include one or more communication ports 31 and 32 (only one communication port is shown with each of such CPUs). The normal unit 4 includes a normal vital cut off relay (VCOR) 33 including an input 34 controlled by one of its vital outputs 35 and an output 36 to one of the vital inputs 37 of the standby unit 6. The standby unit 6 includes a standby VCOR 38 including an input 39 controlled by one of its vital outputs 40 and an output 41 to one of the vital inputs 42 of the normal unit 4.

For each of the normal and standby units 4 and 6, the health routine 8 provides a health status 44N and 44S, respectively, after communication is established with the other one of such units through the communication channel 30. Also, the synchronization routine 10 for each of the normal and standby units 4 and 6 provides a synchronization status 46N and 46S, respectively, through the communication channel 30 with the other one of such units. The normal and standby application routines 12 output the vital outputs 21N and 21S through the vital output boards 20 when the synchronization status 46N and 46S is set and, also, input the vital inputs 19N and 19S, respectively, from the vital input boards 18 regardless of the state of the corresponding synchronization status.

The standby unit 6, which is determined as being standby whenever the vital input 28 (*i.e.*, from the constant low input 26) of such unit is set low, also includes a reset routine 48, which resets the standby CPU 16. The normal unit 6 may include a reset routine (not shown), although that routine is disabled by the vital input 24 (*i.e.*, from the constant high input 22).

As shown with the vital outputs 50,52, an output mechanism 54 (*e.g.*, including diodes 56,58) is provided to output from some of the vital outputs of the normal unit 4 and from some of the vital outputs of the standby unit 6. The output mechanism 54 provides a common output 60 to a suitable output device (not shown). As discussed below, both of the units 4,6 are operating and capable of outputting through the output mechanism 54 to a single output device (not shown) for each pair of the vital outputs 50,52. The output mechanism 54 provides a vital OR circuit having a first input (*i.e.*, the anode of diode 56) from the vital output 50 of the normal unit 4, a second input (*i.e.*, the anode of diode 58) from vital output 52 of the standby

unit 6, and an output (*i.e.*, the common cathodes of diodes 56,58) adapted for electrical connection to a single output device (not shown).

In addition to the serial communication channel 30, an external, handwired synchronization mechanism 61 for exchanging normal and standby synchronization status may be applied between the normal and standby units 4 and 6. A vital output 62 of the normal unit 4 is electrically connected by a suitable conductor 64 to a vital input 66 of the standby unit 6. A vital output 68 of the standby unit 6 is electrically connected by a suitable conductor 70 to a vital input 72 of the normal unit 4.

Example 2

There are three ways a vital Microlok®, such as the ICS 2 of Figure 1, can communicate: serial communication to a non-vital Microlok® (not shown), serial communication between the normal/standby pair 4,6, and vital input and output boards 18,20. Since the communication to the non-vital Microlok® is by definition "non-vital", a breakdown in this communication will have no safety concerns. The serial communication between the normal/standby pair 4,6 utilizes a HEALTH bit (SL.OUT.HEALTH and SL.IN.HEALTH) that constantly monitors the serial link or channel 30. The vital input and output boards 18,20 are tied together. The inputs 19 to the vital input boards 18 are simply paralleled and the outputs 50,52 of the vital output boards 20 are "ORed" together with the diodes 56,58 (since both units 4,6 are preferably permitted to output at all times), or through some type of "Vital OR Gate" (if the user desires the standby unit 6 to suppress its outputs 21S). This means that barring a broken wire (which would result in a failsafe condition), both units 4 and 6 will receive the same inputs 19 and, since they have the same logic equations, produce the same outputs 21N,50 and 21S,52, respectively. Additional information on the outputs 21N,21S,50,52 is provided under the heading Outputs, below.

The application software or routine 12 addresses the hot standby issue with the above in mind. There is no attempt to actively synchronize all bits at all times, and there is no suppression of the outputs 21N,21S,50,52 from either unit 4,6 until a failure or disagreement is detected (although it is possible to suppress the standby outputs 21S,52 if desired by the user). If the outputs of the standby unit 6 are optionally suppressed, then a transfer from the normal unit 4 to the standby unit 6

would occur in a relatively short time, although the transfer would not necessarily be "hot standby". Since the only purpose of the standby unit 6 is to provide a hardware backup for the normal unit 4, the normal unit 4 is considered "boss". If there is a disagreement between the two units 4,6, then the normal unit 4 will always reset the standby unit 6, or the standby unit 6 will reset itself, but the standby unit 6 can never reset the normal unit 4. This is necessary since there is no way of determining which unit 4,6 is correct, but only that they are not in agreement. The only way the normal unit 4 will shut down leaving the standby unit 6 in control is if the normal unit 4 senses an internal failure and takes itself offline. Also, if either unit 4 or 6 is reset, then all of its outputs 21N,50 or 21S,52 are suppressed until they are verified to be in synchronization with the unit currently online.

Preferably, both units 4,6 are permitted to output at all times, in order to provide hot standby operation, and in order to detect a shorted diode 56,58 (Figure 2) from the respective units 4,6, when the outputs of such units are both high. Preferably, the vital output boards 20 include suitable circuitry (not shown), which periodically outputs a low pulse of suitable duration when the corresponding vital output is high, and which periodically outputs a high pulse of suitable duration when the corresponding vital output is low. That circuitry, in turn, monitors the corresponding vital output, in order to verify that the low pulse goes low and that the high pulse goes high. If, for example, the diode 58 of the standby unit 6 is shorted when the standby output 52 is high, then the low pulse from the standby unit 6 would not occur at the common output 60 since the normal output 50 is high and, thus, the common output 60 is driven high. In that manner, for example, the standby unit 6 detects the shorted diode 58, when the outputs of the units 4,6 are both high.

Since each unit 4,6 is wired and programmed virtually identical as it would be if it were a stand-alone unit, this system 2 is very easy to implement. The only modifications needed to produce the hot standby feature are slight modifications in the hardware (*e.g.*, vital inputs, vital outputs, serial communication, vital cut off relay (VCOR) verification and Normal unit bit) and application program.

Inputs

Assuming both units 4,6 are housed by the same rack (not shown), all inputs 19 are single runs to the rack (*e.g.*, most likely to a weidmuller). The inputs 19 are then fed in parallel to each unit 4,6.

Outputs

With both units 4,6 actively producing outputs 21N,50,21S,52 at all times, the outputs, such as 50 and 52, need only be "ORed" together through the pair of diodes 56 and 58 (Figure 2), or through a series of diodes 75 and 78 (*e.g.*, needed to meet relatively more stringent reliability specifications) (Figure 3). As shown in Figure 3, an improved reliability output configuration (as compared to the output mechanism 54 of Figures 1 and 2), includes a first diode array 75 having an input 76 and an output 77 and a second diode array 78 having an input 79 and an output 80. The outputs 77,80 of the first and second diode arrays 75,78, respectively, are adapted for electrical connection at 82 to a single output device (not shown). The input 76 of the first diode array 75 is electrically connected to the vital output 50 of the normal unit 4 of Figure 1. The input 79 of the second diode array 78 is electrically connected to the vital output 52 of the standby unit 6 of Figure 1.

As shown with the first diode array 75, each of the arrays 75,78 includes a first pair 84 of series-connected diodes and a second pair 86 of series-connected diodes. The first pair 84 are electrically connected in parallel with the second pair 86. The first and second pairs 84,86 have a pair of anodes as the input 76 of the corresponding diode array 75. The first and second pairs 84,86 have a pair of cathodes as the output 77 of the corresponding diode array 75.

However, the diodes of Figures 2 and 3 cannot be used if the user requires that the outputs of the standby unit 6 be suppressed if the normal unit 4 is online. The reason for this is that although a Microlok® can reliably detect a shorted diode when both units 4,6 are active, it cannot reliably detect a shorted diode on the normal unit 4 if the standby unit 6 is not producing outputs (see Scenarios 8 and 9 of Tables 8 and 9, respectively, below). If the outputs 52 of the standby unit 6 are suppressed, then the outputs 50,52 must be "ORed" through some type of "Vital OR Gate" (not shown) which will not allow the possibility of shorting. Union Switch & Signal's Isolation Module can serve this purpose, but it is not cost effective.

Alternatively, a scaled down version (not shown) could be developed which would provide the necessary protection and be smaller and less expensive.

Serial Communication

Continuing to refer to Figure 1, two vital serial communications ports (*e.g.*, COM1 and COM2) (only single ports 31,32 are shown) are utilized on each unit 4,6. This still leaves either two non-vital or one vital and one non-vital port (*e.g.*, COM3 (not shown) can be set up as either vital or non-vital) for links to other units (not shown). It is possible to implement the system 2 using only one communication port of each unit 4,6, but that would require a variation in the application program in each unit. This can easily be accomplished, but it is unnecessary unless more than two communication ports are required for links to other units.

VCOR Verification

In order to constantly monitor the condition of the other unit, each of the units 4,6 must have a front contact 41,36 of the other unit's VCOR relay 38,33 connected to an input 42,37, respectively, of one of its vital input boards, such as 18.

Normal Unit Bit

Since the same application program, such as 12, is uploaded into both of the units 4,6, each of such units must have one input 24,28, respectively, that is used for identification. This input 24 (*e.g.*, on vital input board 18) must be constantly high in the normal unit 4 and constantly low in the standby unit 6. The application routine 12 uses these constant bit states (Normal - high, Standby - low) in portions of the assign statements, as discussed below, that require different operating characteristics for the normal unit 4 than for the standby unit 6.

Application Program

Any application program designed for a stand-alone unit (not shown), such as one of units 4,6, can be changed to a hot standby application simply by adding three logic systems and modifying all output bits to be one of three types. Also, if external Lock relays (not shown) are not utilized, then the internal Lock bits must be modified as if they were outputs.

Logic Systems

The three logic systems: (1) Synchronization 10, (2) Health 8, and (3) Reset 48 serve to restrict, maintain, and protect the operation of the hot standby

system. More detailed explanations of the bits that comprise these systems can be found in the example test programs in the Appendices.

Synchronization

The Synchronization system of synchronization routine 10 restricts the corresponding unit 4,6 from producing outputs 50,52 if the other unit 6,4, respectively, is already online and the output states of the units 4,6 disagree. Once the corresponding unit achieves synchronization, it is permitted to produce its corresponding vital outputs 50,52 and the synchronization routine 10 is not utilized until the unit is reset and attempts to come back on line. This system is equally functional in both the normal and standby units 4,6.

The synchronization routine 10 employs the following bits:

STAND.ALONE.SYNC.DELAY is a slow set bit (*e.g.*, a suitable delay is provided before setting the bit; no delay is provided before clearing the bit) that provides a 1 second delay for the corresponding unit 4,6 to stabilize before the other unit's VCOR 38,33, respectively, is referenced.

STAND.ALONE.SYNC sets the SYNC bit, below, or synchronization status 46N,46S if STAND.ALONE.SYNC.DELAY is set and the other unit's VCOR 38,33, respectively, is down.

SYNC.WAIT is a slow set bit which forces the unit 4,6 coming online to wait until serial communication is stabilized over the communication channel 30 before attempting to synchronize.

SYNC is the controlling bit. When the unit 4,6 coming online is synchronized with the other unit currently online the SYNC bit is set.

Health

The Health system of health routine 8 is verified by the constant exchange of the HEALTH bit over the serial communication channel 30. When the normal unit's VCOR 33 is picked, the HEALTH bit is required for the standby unit 6 to stay online. Without the HEALTH bit verifying that serial communication is stable, the standby unit 6 is reset by its reset routine 48. This ensures that if communication is lost, one unit (*i.e.*, the standby unit 6) is taken offline. Though this system is primarily utilized in the standby portions of the assign statements, as

discussed below, the normal unit 4 also uses a HEALTH.WAIT bit to maintain its Restricted bits, as discussed below, while the standby unit 6 is coming online.

The health routine 8 employs the following bits:

HEALTH.WAIT.DELAY is a slow clear bit (*e.g.*, a suitable delay is provided before clearing the bit; no delay is provided before setting the bit) that is set when the other unit's VCOR 38,33 is picked but serial communication is not yet established over the communication channel 30. The function of this bit is to set HEALTH.WAIT, below, and maintain it until either SL.IN.HEALTH, below, is received from the other unit or time expires.

HEALTH.WAIT is a slow set bit that sets 1 second after HEALTH.WAIT.DELAY. This bit clears when SL.IN.HEALTH is received from the other unit or time expires and HEALTH.WAIT.DELAY clears.

SL.OUT.HEALTH is the serial bit that the unit 4,6 sends to the other unit, 6,4, respectively.

SL.IN.HEALTH is the serial bit that the unit 4,6 receives from the other unit 6,4, respectively.

Reset

The Reset system of standby reset routine 48 protects the pair's vital functions by forcing the standby unit 6 to reset when there is a disagreement between the units 4,6. This system is always active in the standby unit 6 if the normal unit's VCOR 33 is picked.

The reset routine 48 employs the following bits:

SYS.RESET is a slow set bit that is only operational in the standby unit 6. When this bit sets, the standby unit 6 resets.

SL.OUT.RESET is sent (*e.g.*, over the communication channel 30) from the normal unit 4 to the standby unit 6 when the normal unit 4 determines there is a disagreement and wants the standby unit 6 to reset.

SL.IN.RESET is the bit the standby unit 6 receives over the communication channel 30 when the normal unit 4 sends SL.OUT.RESET.

GROUP.XX.RESET type bits are groups of individual reset bits that are used to simplify the SYS.RESET assign statement in the application routine 12 and eliminate the need for timers on all individual reset bits.

Bit Types

There are three types of bits: Unrestricted, Half Restricted, and Restricted. All three of these types may be utilized to ensure that the hot standby operates safely, but does not waste system resources on unnecessary tasks.

These three types of bits have the following in common: (1) if the other unit's VCOR 38,33 is down, then the unit 4,6, respectively, will produce the output 50,52 whenever the assign statement is satisfied; (2) if the other unit's VCOR 38,33 is up, then the unit 4,6, respectively, must also receive serial communication over the communication channel 30, (a) Unrestricted bits require a generic health bit, (b) Half Restricted bits require a bit verification from the normal unit 4 to the standby unit 6, and (c) Restricted bits require bit verification to and from both units 4,6; (3) if the other unit 4,6 is in control, then the unit 6,4 being brought online cannot produce any outputs 52,50, respectively, until it is in SYNC; and (4) if both units 4,6 are online and any bit states disagree for a selected period of time, then either the normal unit 4 will reset the standby unit 6 or the standby unit 6 will reset itself.

Alternatively, with minor optional modifications, the standby unit's outputs 21S,52 can be suppressed, for example, when the normal unit's VCOR 33 is picked.

Unrestricted

These bits require no bit specific serial communication between the units 4,6 in order to produce an output, such as 21N,21S,50,52; therefore, they are the fastest and should always be utilized whenever possible. These bits should never be used for signal lighting or Locks.

Half Restricted

These bits are unrestricted in the normal unit 4, but restricted in the standby unit 6. The standby unit 6 cannot produce the output 21S,52 until it receives verification (via serial communication over the communication channel 30) that the normal unit 4 has also satisfied the assign statement. This type of bit is specifically designed for signal lighting. If these bits are out of sync, then it can only be that the normal unit 4 has the aspect lit and the standby unit 6 does not. In this event, the standby unit 6 is reset, and the signal aspect does not change.

Restricted

These bits are restricted in both the normal and the standby units 4,6. Neither unit 4,6 can produce the output 50,52 until it receives verification (via serial communication over the communication channel 30) that the other unit 6,4, respectively, has also satisfied the assign statement. This type of bit is the slowest due to the amount of serial communication involved. It is specifically designed for Locks. The bit cannot be set (unlocked) until both units 4,6 satisfy the assign statement and it will be cleared (locked) immediately at any time the units 4,6 do not agree. There are two considerations concerning this type of bit: (1) if locking is performed without the use of external Lock relays, then the internal variables will require this configuration; and (2) if the response time is too long due to the use of serial communication, then the verification will need to be passed via vital input and output boards 18,20.

Example 3

Both units 4,6 were housed in a cabinet and shared the same (not shown) power supply, such as 14. The serial communication between the units 4,6 was accomplished with a cable 30 from the normal CPU 16 to the standby CPU 16. The communication cable 30 tied normal COM1 to standby COM2, and normal COM2 to standby COM1 (only single communication ports 31,32 are shown in Figure 1). Inputs 19 were paralleled to both units 4,6, but the wires going to the standby unit 6 were clipped together and could be disconnected. The outputs 50,52 were "ORed" together with diodes 56,58, respectively, as shown in the basic configuration example of Figures 1 and 2.

Both units 4,6 were uploaded with an application program 12.

The following references were used for testing purposes: (1) "VCOR picked" was referenced from the lighting of the VCOR indication on the power supply board 14; (2) serial communication was referenced from the COM indications (A,B,C,D, and E) on the CPU board 16; (3) outputs were referenced from the indications on the vital output board 20; and (4) for the purpose of testing, the following reference bits were not considered to be vital outputs: OUT 7 - SYNC.WAIT, OUT 8 - SYNC, OUT 9 - HEALTH.WAIT.DELAY, OUT 10 -

HEALTH.WAIT, OUT 12 - SL.OUT.04, OUT 13 - SL.IN.04, OUT 14 - OUT.RESET, OUT 15 - IN.RESET, and OUT 16 - COMALT.

Tables 1-9, below, show different test scenarios for Example 3, above.

Scenario 1	Unit RESET while the other unit is offline (CPU pulled)
Purpose	To verify that each unit can operate as a stand-alone unit
Attempts	5 (for each unit)
Result	Unit came online and produced outputs: • Time from RESET to picking of VCOR = 14 sec. • Time from RESET to outputs = 15 sec. • Time from RESET to attempted serial communication = 25 sec.
Comment	• The times were the same for both the normal and standby units

Table 1

Scenario 2	Unit RESET while the other is online, communicating, and producing outputs
Purpose	To verify that each unit can be brought online without any interruption of controlling unit
Attempts	5 (for each unit)
Result	Unit came online and produced outputs: • Time from RESET to picking of VCOR = 14 sec. • Time from RESET to serial communication = 25 sec. • Time from RESET to outputs = 30 sec.
Comment	• The times were the same for both the normal and standby units • No change in outputs occurred in the other unit

Table 2

Scenario 3	Power up both units simultaneously
Purpose	To verify that there is no circular logic, which would prevent the units from coming online simultaneously
Attempts	5
Result	Both units powered up and produced outputs: • Time from power to picking of VCOR = 18 sec. • Time from power to serial communication = 29 sec. • Time from power to both unit's outputs = 35 sec.
Comment	• This functioned as expected • No interruption of outputs occurred in either unit

Table 3

Scenario 4	• Both units online • Remove IN.02 bit from the standby unit
Purpose	To prove that the normal unit will RESET the standby unit if there is a disagreement in bit states, and that there is no danger in allowing the standby unit to attempt to come back online after it is RESET
Attempts	• 1 disconnect of input • Standby unit RESET 34 times • 20 minutes time
Result	Standby unit RESET and continued to cycle: • Time from removal of input to first RESET = 10 sec. • Time from removal of input to first VCOR pick = 23 sec. • Time from removal of input to first establishment of serial communication with normal unit = 35 sec. • Time from removal of input to second RESET = 38 sec.
Comment	• Standby unit continued to cycle, resetting every 15 seconds after the VCOR picked • The standby unit never produced any outputs • The normal unit's outputs were never interrupted • After approximately 30 minutes IN.02 was restored to standby unit and after the completion of its current reset, it came back online and produced outputs

Table 4

Scenario 5	• Both units online • Remove IN.02 bit from the standby unit • Reset normal unit • Wait 1 minute 30 seconds • Restore IN.02 bit to the standby unit
Purpose	To prove that the synchronization is required for the normal unit to come online if the standby unit is in control and that the normal unit cannot RESET the standby unit as the normal unit comes online
Attempts	5
Result	Normal unit did not produce outputs until after IN.02 was restored to the standby unit: • Time from RESET to VCOR pick = 14 sec. • Time from RESET to serial communication = 25 sec. • Time from RESET to IN.02 restored to standby unit = 1 min. 30 sec. • Time from RESET to output = 1 min. 30 sec.

Table 5

5

Scenario 6	• Both units online • RESET unit while IN.02 and IN.04 are constantly toggled (approximately 4 times a second)
Purpose	To prove that the flashing bits or several bits changing state will not hinder the unit from synchronizing and coming online
Attempts	5 (for each unit)
Result	• Both units came online in the usual timeframe
Comment	• The controlling unit's outputs were never interrupted

Table 6

Scenario 7	• Both units online • Remove OUT.02 and OUT.04 set, and bit OUT.03 clear • Reset unit • Wait 5 seconds • RESET other unit
Purpose	To prove that either unit will produce outputs immediately if the other unit's VCOR is down
Attempts	5 (for each unit)
Result	• The unit RESET first came online and immediately produced outputs • The unit RESET second had to synchronize before it produced outputs
Comment	• The controlling unit's outputs were never interrupted

Table 7

Scenario 8	• Both units online • Place a short across one output diode
Purpose	To prove that a shorted diode can be detected by the normal and standby units
Attempts	• Twice on each diode of OUT.02, OUT.03 and OUT.04 with the outputs high • Twice on each diode of OUT.02, OUT.03 and OUT.04 with the outputs low • A total of 24 tests
Result	• When the outputs were high, both units detected the short in 9-10 seconds and RESET • When the outputs were low the normal unit detected the short within 1 second and RESET • When the outputs were low the standby unit did not detect the short. All outputs were shorted for at least 1 minute and one was allowed to remain shorted for over 8 minutes and the unit still failed to detect it. After the elapsed time, the output was toggled high and the short was detected within 10 seconds.
Comment	• The conclusion is that the normal and standby units can consistently detect a shorted diode within 10 seconds if both outputs are high • The unit without the shorted diode maintained its output without interruption • Since some properties would require the standby unit's outputs to be suppressed if the normal unit's VCOR is picked, the question still remains if the normal and standby units can detect a shorted diode if the outputs are not the same (both high or both low). See Scenario 9 (Table 9).

Table 8

Scenario 9	• Install modified program in both the normal and standby units, which program suppresses the standby unit's outputs if the normal unit's VCOR is picked • Bring both units online (the normal unit has outputs the standby unit does not) • Place a short across one output diode
Purpose	To determine if shorted diode can be detected by the normal and standby units if the outputs are not the same (both high or both low)
Attempts	• Twice on each diode of OUT.02, OUT.03 and OUT.04 • A total of 12 tests
Result	• When the diode of the standby unit was shorted the standby unit detected the short within 1 second and RESET • When a diode of the normal unit was shorted the normal unit did not detect the short. All outputs were shorted for at least 1 minute and one was allowed to remain shorted for over 8 minutes and the unit still failed to detect it. After the elapsed time, the normal unit was RESET and when it attempted to come back online (its outputs low and the standby unit's outputs high) it detected the short and continually RESET.
Comment	• The conclusion is that the normal and standby units cannot detect a shorted diode when it is on the high output and the other output is low • If the standby unit's outputs must be suppressed when the normal units VCOR is picked then diodes are not a viable option

Table 9

Example 4

Tables 10 and 11, below, show the configuration of the normal and standby test units, respectively, of Example 3, above.

NORMAL UNIT				
BOARD	SLOT	PART#	SERIAL #	REVISION
CPU	18	N17003401	1201028	2
Vital Input - 16	13	N17001001	0998048	2
Vital Output - 16	15	N17000501	0898006	2
Power Supply	6	N16600301	1998022	4

Table 10

STANDBY UNIT				
BOARD	SLOT	PART#	SERIAL #	REVISION
CPU	18	N17001301	3100010	7
Vital Input - 16	13	N17001001	0998033	2
Vital Output - 16	15	N17000501	0898005	2
Power Supply	10	N451810750	2498001	2

Table 11

Example 5

Figure 4 shows the signal interconnections between two normal and standby units 4′,6′. A vital "OR" circuit 54′ outputs from the vital outputs 21N′,21S′ of respective normal and standby units 4′,6′. These units 4′ and 6′ are similar to the units 4 and 6 of Figure 1, except that they include two communication ports 31O and 31I, and 32O and 32I, respectively, as part of a communication channel 30′. In this example, there are three vital outputs, which are output by the assign statements (as shown in the Appendices) of the application routines 12 of the normal and standby units 4′,6′. Although three vital outputs are shown for each unit, one, two or more

than three vital outputs may be employed. The three vital outputs 21N′,21S′ include an Unrestricted output (OUT.02), a Half Restricted output (OUT.H.03) and a Restricted output (OUT.L.04). The assign statements qualify these outputs by employing the normal and standby states of such units 4′ and 6′, as defined by the constant high (normal) input 22 and the constant low (standby) input 26, respectively, of Figure 1. In turn, the vital "OR" circuit 54′ outputs three vital outputs 60′ (OUT2, OUT3, OUT4) to three corresponding output devices (not shown).

With respect to the normal unit 4′, an output communication path 88 from the normal unit 4′ to the standby unit 6′ is provided by the normal output communication port 31O and the standby input communication port 32I. Also, an input communication path 90 to the normal unit 4′ from the standby unit 6′ is provided by the standby output communication port 32O and the normal input communication port 31I. The routines 8,10,12 (Figure 1) of the normal unit 4′ output six bits 92 (SL.OUT.02, SL.OUT.H.03, SL.OUT.L.04, SL.OUT.HEALTH, SL.OUT.RESET, SL.OUT.SYNC), which bits 92 are input as bits 94 (SL.IN.02, SL.IN.H.03, SL.IN.L.04, SL.IN.HEALTH, SL.IN.RESET, SL.IN.SYNC) by the routines 8,10,12,48 (Figure 1) of the standby unit 6′. In a similar manner, the routines 8,10,12 of the standby unit 6′ output three bits 96 (SL.OUT.02, SL.OUT.H.03, SL.OUT.L.04), which bits 96 are input as bits 98 (SL.IN.02, SL.IN.H.03, SL.IN.L.04) by the routines 8,10,12 of the normal unit 4′.

The hot standby method and apparatus disclosed herein is organized in such a way that it can easily be incorporated into any Microlok® application program, such as application routine 12, in order to produce a hot standby.

While specific embodiments of the invention have been described in detail, it will be appreciated by those skilled in the art that various modifications and alternatives to those details could be developed in light of the overall teachings of the disclosure. Accordingly, the particular arrangements disclosed are meant to be illustrative only and not limiting as to the scope of the invention which is to be given the full breadth of the claims appended and any and all equivalents thereof.

Example 6

Appendix A

MICROLOK_II PROGRAM;

INTERFACE

LOCAL

BOARD: VO_SLOT_J15

ADJUSTABLE ENABLE:	1
TYPE:	OUT16

OUTPUT:

/* Only OUT.02, OUT.H.03, and OUT.L.04 simulate true outputs. The remaining bits are only used to give an indication on the output board for testing purposes. */

SPARE,	OUT.02,	OUT.H.03,	OUT.L.04,
SPARE,	SPARE,	OUT.SYNC.WAIT.07,	OUT.SYNC.08,
OUT.HEALTH.WAIT.DELAY.09,	OUT.HEALTH.WAIT.10,	SPARE,	OUT.OUT.L.04.12,
OUT.IN.L.04.13,	OUT.OUT.RESET.14,	OUT.IN.RESET.15,	OUT.COMALT.16;

BOARD: VI_SLOT_J13

ADJUSTABLE ENABLE:	1
TYPE:	IN16

INPUT:

/* In the Normal unit, NORMAL (bit 16) is energized from a constant source. It must be high in the Normal unit and low in the Standby unit. */

VCOR,	IN.02,	IN.03,	IN.04,
SPARE,	SPARE,	SPARE,	SPARE,
SPARE,	SPARE,	SPARE,	SPARE,
SPARE,	SPARE,	SPARE,	NORMAL;

COMM

```
/*      Two COM ports are used so that the same software can be used in both units.      */
```

LINK: HOT_MASTER

```
ADJUSTABLE      ENABLE:              1
                PROTOCOL:            MICROLOK.MASTER
ADJUSTABLE      POINT.POINT:         1;
ADJUSTABLE      PORT:                1;
ADJUSTABLE      BAUD:                19200;
ADJUSTABLE      STOPBITS:            1;
ADJUSTABLE      PARITY:              NONE;
ADJUSTABLE      KEY.ON.DELAY:        12;
ADJUSTABLE      KEY.OFF.DELAY:       12;
ADJUSTABLE      STALE.DATA.TIMEOUT:  3:SEC;
ADJUSTABLE      POLLING.INTERVAL:    50:MSEC;
ADJUSTABLE      MASTER.TIMEOUT:      100:MSEC;
```

```
ADDRESS: 1

                ADJUSTABLE ENABLE:   1
```

OUTPUT:

```
/*      SL.OUT.02, SL.OUT.03, and SL.OUT.04 represent all output bits. If the
        Microlok had two 16 bit Vital Output Boards (and all bits were used) then
        there would be 32 bits listed.
        SL.OUT.HEALTH, SL.OUT.RESET, and SL.OUT.SYNC are the only extra
        bits required for Hot Standby operation.                                    */
```

```
SL.OUT.02,          SL.OUT.H.03,        SL.OUT.L.04,

SL.OUT.HEALTH,      SL.OUT.RESET,       SL.OUT.SYNC;
```

LINK: HOT_SLAVE

```
ADJUSTABLE      ENABLE:              1
                PROTOCOL:            MICROLOK.SLAVE
ADJUSTABLE      POINT.POINT:         0;
ADJUSTABLE      PORT:                2;
ADJUSTABLE      BAUD:                19200;
ADJUSTABLE      STOPBITS:            1;
ADJUSTABLE      PARITY:              NONE;
ADJUSTABLE      KEY.ON.DELAY:        12;
ADJUSTABLE      KEY.OFF.DELAY:       12;
ADJUSTABLE      STALE.DATA.TIMEOUT:  3:SEC;
```

```
ADDRESS: 1

                ADJUSTABLE ENABLE:   1
```

INPUT:

/* SL.IN.02, SL.IN.03, and SL.IN.04 represent all input bits from the other unit. If the Microlok had two 16 bit Vital Output Boards (and all bits were used) then there would be 32 bits listed. SL.IN.HEALTH, SL.IN.RESET, and SL.IN.SYNC are the only extra bits required for Hot Standby operation. */

SL.IN.02, SL.IN.H.03, SL.IN.L.04,

SL.IN.HEALTH, SL.IN.RESET, SL.IN.SYNC;

BOOLEAN BITS

SYS.RESET, GROUP.01.RESET, GROUP.02.RESET, GROUP.03.V.RESET,

OUT.RESET.02, OUT.H.RESET.03, OUT.L.RESET.04,

SYNC, SYNC.WAIT, STAND.ALONE.SYNC.DELAY, STAND.ALONE.SYNC,

OUT.02.SYNC, OUT.H.03.SYNC, OUT.L.04.SYNC,

HEALTH.WAIT.DELAY, HEALTH.WAIT,

COMALT,

SL.IN.H.03.D, SL.IN.L.04.D;

TIMER BITS

/* SL.OUT.RESET is sent from the Normal unit to the Standby unit when the Normal unit determines there is a disagreement in bit states. It is delayed to allow the Standby unit time to synchronize. The exact setting for this bit is based on the needs of each application. It should be as short as possible without effecting reliability. */

SL.OUT.RESET:	SET = 3:SEC	CLEAR = 0:SEC;

/* SYS.RESET is an internal bit that RESETS the Standby unit if it is out of synchronization with the online Normal unit. It is slightly delayed to insure that the Standby unit does not falsely reset. The exact setting for this bit is based on the needs of each application. It should be as short as possible without effecting reliability. */

SYS.RESET:	SET = 3:SEC	CLEAR = 0:SEC;

/* The GROUP.RESET bits represent groups of individual bit resets. They are slightly delayed to insure that the Standby unit does not reset falsely. GROUP.03.V.RESET contains bits that are "more vital" such as Switch Locking or Route Locking, therefore they are given a shorter reset time. The exact setting for these bits is based on the needs of each application. They should be as short as possible without effecting reliability. */

GROUP.01.RESET:	SET = 3:SEC	CLEAR = 0:SEC;
GROUP.02.RESET:	SET = 3:SEC	CLEAR = 0:SEC;
GROUP.03.V.RESET:	SET = 1:SEC	CLEAR = 0:SEC;

/* HEALTH.WAIT.DELAY is an internal bit that allows the Standby unit to maintain its outputs while the Normal unit is brought online. It fills in the gap between the time the Normal's VCOR picks and communication between the pair is established. It is set for 20 seconds because it takes approximately 15 seconds for a unit to establish serial communication after the VCOR is picked. */

HEALTH.WAIT.DELAY:	SET = 0:SEC	CLEAR = 20:SEC;

/* HEALTH.WAIT shortens the effect of HEALTH.WAIT.DELAY to 1 second after serial communication is established. HEALTH.WAIT is used in all output bit assign statements. */

HEALTH.WAIT:	SET = 0:SEC	CLEAR = 1:SEC;

/* STAND.ALONE.SYNC.DELAY is a slow set bit that allows the unit to stabilize before VCOR is referenced for SYNC. */

STAND.ALONE.SYNC.DELAY:	SET = 1:SEC	CLEAR = 0:SEC;

/* SYNC.WAIT is a slow set internal bit that allows serial communication to stabilize after the unit is powered up before synchronization is verified. It should always be set for 5 seconds or longer. */

SYNC.WAIT:	SET = 5:SEC	CLEAR = 0:SEC;

/* SL.IN.H.03.D and SL.IN.L.04.D are test bits used to simulate the delay in serial communications between the units. */

SL.IN.H.03.D:	SET = 1:SEC	CLEAR = 0:SEC;
SL.IN.L.04.D:	SET = 1:SEC	CLEAR = 0:SEC;

CONSTANTS BOOLEAN

ONE	=	1;
ZERO	=	0;

CONFIGURATION

SYSTEM

ADJUSTABLE	DEBUG_PORT_ADDRESS:	1;
ADJUSTABLE	DEBUG_PORT_BAUDRATE:	9600;
ADJUSTABLE	LOGIC_TIMEOUT:	2:SEC;
ADJUSTABLE	DELAY_RESET:	3:SEC;

LOGIC BEGIN

/* UTILITY BITS */

ASSIGN	ONE	TO	CPS.ENABLE;
ASSIGN	ONE	TO	STAND.ALONE.SYNC.DELAY ;
ASSIGN	ONE	TO	SL.OUT.HEALTH;
ASSIGN	SYS.RESET	TO	RESET;

/* RESET BITS */

SYS.RESET is a slow set bit that only functions in the Standby unit.
The bits in the assign statement function as follows:

- ~NORMAL insures that only the Standby unit can be RESET.
- VCOR insures that the unit will only RESET if the Normal unit is online.
- SL.IN.RESET comes from the Normal unit and forces the Standby unit to RESET.
- ~SL.IN.HEALTH insures the Standby unit will RESET itself if serial communication is lost between the units.
- ~HEALTH.WAIT insures the Standby unit will not RESET itself before serial communication is established when the Normal unit is coming online.
- SL.IN.SYNC insures that the Standby unit will only RESET itself if the Normal unit is in sync.
- ~SYNC insures that the Standby unit will RESET itself if both units are powered up simultaneously and do not achieve synchronization. This permits the Normal unit to take control.
- SYNC.WAIT delays the RESET until the Standby unit has an opportunity to verify synchronization with the Normal unit.
- GROUP.01.RESET, GROUP.02.RESET, and GROUP.03.V.RESET are groups of individual reset bits.

/* */

ASSIGN	(~NORMAL * VCOR) * (SL.IN.RESET + ((~SL.IN.HEALTH * ~HEALTH.WAIT) + ((SL.IN.SYNC + (~SYNC * SYNC.WAIT)) * (GROUP.01.RESET + GROUP.02.RESET + GROUP.03.V.RESET))))	TO	SYS.RESET;

SL.OUT.RESET is a slow set bit that is sent from the Normal unit to the Standby unit when any output bit is out of sync. It is primarily controlled by the GROUP.RESET bits, however, the SYNC bit is also required so that the Normal unit cannot reset the Standby unit if the Normal is being powered up and cannot achieve synchronization with the Standby.

/* */

ASSIGN	(NORMAL * SYNC) * (GROUP.01.RESET + GROUP.02.RESET + GROUP.03.V.RESET)	TO	SL.OUT.RESET;

/* GROUP.01.RESET, GROUP.02.RESET, and GROUP.03.V.RESET are groups of individual reset bits (though only one RESET bit is assigned to each for testing). The individual reset bits are grouped together to simplify the SYS.RESET equation and to allow for a longer time delay for non-synchronous situations which may be caused by serial communication delays. Three groups are used for testing but the maximum number of groups is unlimited. Multiple groups should be used to limit the number of bits so that continuous changes of bit states will not by misinterpreted as a non-synchronous condition.
GROUP.03.V.RESET represents groups of bits that are "more vital" such as Switch Locks and Route Locks. This group is given the absolutely shortest time delay possible while still maintaining reliability. */

ASSIGN	OUT.RESET.02	TO	GROUP.01.RESET;
ASSIGN	OUT.H.RESET.03	TO	GROUP.02.RESET;
ASSIGN	OUT.L.RESET.04	TO	GROUP.03.V.RESET;

/* SYNCHRONIZATION BITS */

/* SYNC suppresses all outputs of the unit being brought online until they are verified to be synchronous with the unit currently in control or the other unit's VCOR is down. Once it is set it is stuck high until the unit is powered down. SL.OUT.SYNC is sent out to the other unit. It is utilized by the Standby unit in the SYS.RESET assign statement. */

ASSIGN	SYNC + STAND.ALONE.SYNC + (OUT.02.SYNC * OUT.H.03.SYNC * OUT.L.04.SYNC)	TO	SYNC, SL.OUT.SYNC;

/* SYNC.WAIT is a slow set bit that suppresses verification of bits in the unit being brought online until it is powered up and both the unit and the serial communication link are stable. Once it is set it is stuck high until the unit is powered down. */

ASSIGN	SYNC.WAIT + (VCOR * SL.IN.HEALTH)	TO	SYNC.WAIT;

/* STAND.ALONE.SYNC is an internal bit that will set the SYNC bit one second after the unit is powered up if the other unit's VCOR is down. */

ASSIGN	~VCOR * STAND.ALONE.SYNC.DELAY	TO	STAND.ALONE.SYNC;

/* **HEALTH BITS** */

/* HEALTH.WAIT and HEALTH.WAIT.DELAY allow the Standby unit to maintain its outputs between the time that the Normal unit's VCOR picks and communication is established between the units. HEALTH.WAIT.DELAY is a slow clear bit that sets when the VCOR picks in the unit coming online. HEALTH.WAIT.DELAY sets HEALTH.WAIT which remains high until either HEALTH.WAIT.DELAY expires or serial communication is established. This is necessary to insure that as soon as serial communication is established the HEALTH bit is not able to override any logic. */

ASSIGN	VCOR * ~SL.IN.HEALTH * ~HEALTH.WAIT.DELAY	TO	HEALTH.WAIT.DELAY;
ASSIGN	HEALTH.WAIT.DELAY * ~SL.IN.HEALTH	TO	HEALTH.WAIT;

/* **OUTPUT BITS** */

There are three types of output bits:

1. Unrestricted, represented by OUT.02.

These bits require no bit specific serial communication between the units in order to produce an output; therefore they are the fastest and should always be utilized whenever possible. They should never be used for signal lighting or any type of locking.

2. Half Restricted, represented by OUT.03.

These bits are unrestricted in the Normal unit, but restricted in the Standby. The Standby unit cannot produce the output until it receives verification (via serial communication) that the Normal unit has also satisfied the assign statement. This type of bit is specifically designed for signal lighting. If the bits are out of sync, it can only be that the Normal unit has the aspect lit and the Standby does not. In this event the Standby unit is reset, and the signal aspect will not change.

3. Restricted, represented by OUT.04.

These bits are restricted in both the Normal and the Standby units. Neither unit can produce the output until it receives verification (via serial communication) that the other unit has also satisfied the assign statement. This type of bit is the slowest due to the amount of serial communication involved. It was specifically designed for locking. The bit cannot be set (unlocked) until both units satisfy the assign statement and it will be cleared (locked) immediately at any time that the units do not agree.

The three types of bits have the following in common:

- If the other unit's VCOR is down the unit will produce the output whenever the assign statement is satisfied.
- If the other unit's VCOR is up the unit must also receive serial communication.
  - Unrestricted bits require a generic health bit.
  - Half-Restricted bits require a bit verification from Normal to Standby.
  - Restricted bits require bit verification to and from both units.
- If the other unit is in control, the unit being brought online cannot produce any output until it is in SYNC.
- If both units are online and any bit becomes out of sync for a selected period of time either the Normal unit will reset the Standby or the Standby will reset itself.

```
/*  UNRESTRICTED BITS                                                    */
```

```
/*  The term IN.02 is used for testing purposes. In reality it would be replaced by a
    logic equation. In the term SL.OUT.02, SL stands for Serial Link, and OUT.02
    represents the resulting bit of the satisfied assign statement. SL.OUT.02 is
    immediately sent out serially to the other unit.                     */
```

```
ASSIGN    IN.02    TO    SL.OUT.02;
```

```
/*  OUT.02.SYNC is primarily satisfied by the SL.OUT.02 bit. If the other unit is online
    (VCOR is picked, or in the process of booting up) it is referenced to insure that the
    bit is in the same state. Once it is set it is stuck high until the unit is powered down.
    If the other unit is offline (VCOR down) this bit is bypassed and the SYNC bit assign
    statement is satisfied with STAND.ALONE.SYNC.                        */
```

```
ASSIGN    OUT.02.SYNC +
          (((SL.OUT.02 * SL.IN.02) +
          (~SL.OUT.02 * ~SL.IN.02))
          * VCOR * SYNC.WAIT)    TO    OUT.02.SYNC;
```

/* OUT.02 is the bit that sets the output high on the Vital Output Board. It is primarily satisfied by the SL.OUT.02 bit. In the Normal unit the only other requirement is that the SYNC bit must be set (which it will be unless the Normal is in the process of coming online). The Standby unit requires a serial communication HEALTH bit or HEALTH.WAIT. HEALTH.WAIT is used keep the Standby unit's outputs set between the time the Normal unit's VCOR picks and serial communication is established when the Normal unit is being brought online. Both the Normal and Standby units will immediately set the bit if SL.OUT.02 is high and the other unit is offline (VCOR down). */

ASSIGN	SL.OUT.02 * (SYNC * (NORMAL + (~NORMAL * (SL.IN.HEALTH + HEALTH.WAIT))) + ~VCOR)	TO	OUT.02;

/* OUT.RESET.02 causes the Standby unit to reset if there is a disagreement in the bit state between the units. */

ASSIGN	(SL.OUT.02 * ~SL.IN.02) + (~SL.OUT.02 * SL.IN.02)	TO	OUT.RESET.02;

/* **HALF RESTRICTED BITS** */

/* The logic statement for OUT.H.03 functions the same as OUT.02 above, with one exception. In the statement for OUT.H.03 the generic serial communication HEALTH bit is replaced with the corresponding bit (SL.IN.H.03.D) from the Normal unit. This suppresses the output from the Standby unit until it has been verified that the Normal has also satisfied the assign statement. SL.IN.H.03.D is a slow set bit used for testing to simulate serial communication delays. */

ASSIGN	IN.03	TO	SL.OUT.H.03;
ASSIGN	OUT.H.03.SYNC + (((SL.OUT.H.03 * SL.IN.H.03) + (~SL.OUT.H.03 * ~SL.IN.H.03)) * VCOR * SYNC.WAIT)	TO	OUT.H.03.SYNC;
ASSIGN	SL.OUT.H.03 * (SYNC * (NORMAL + (~NORMAL * (SL.IN.H.03.D + HEALTH.WAIT))) + ~VCOR)	TO	OUT.H.03;
ASSIGN	(SL.OUT.H.03 * ~SL.IN.H.03) + (~SL.OUT.H.03 * SL.IN.H.03)	TO	OUT.H.RESET.03;

/* **RESTRICTED BITS** */

/* The logic statements for OUT.L.04 are the same as OUT.H.03 above, with one exception. In the statement for OUT.L.04 there are no separate variables for Normal or Standby. Both units must have SL.OUT.L.04, be in SYNC, and receive the corresponding bit from the other unit. This suppresses the output from either unit until it has been verified that the other unit has also satisfied the assign statement and immediately drops the output if it loses the verification from the other unit. */

ASSIGN	IN.04	TO	SL.OUT.L.04;
ASSIGN	OUT.L.04.SYNC + (((SL.OUT.L.04 * SL.IN.L.04) + (~SL.OUT.L.04 * ~SL.IN.L.04)) * VCOR * SYNC.WAIT)	TO	OUT.L.04.SYNC;
ASSIGN	SL.OUT.L.04 * (SYNC * (SL.IN.L.04.D + HEALTH.WAIT) + ~VCOR)	TO	OUT.L.04;
ASSIGN	(SL.OUT.L.04 * ~SL.IN.L.04) + (~SL.OUT.L.04 * SL.IN.L.04)	TO	OUT.L.RESET.04;

/* **COMMUNICATION ALERT BIT** */

/*	COMALT may be used to alert central control of any problems in the communication between the Normal and Standby units.	*/

ASSIGN	~SL.IN.HEALTH	TO	COMALT;

/* **FOR TEST PURPOSES ONLY** */

/* **COMMUNICATION DELAY SIMULATORS** */

/*	SL.IN.H.03.D and SL.IN.L.04.D are slow set bits that simulate the possible delay in the serial communication between the Normal and Standby units.	*/

ASSIGN	SL.IN.H.03	TO	SL.IN.H.03.D;
ASSIGN	SL.IN.L.04	TO	SL.IN.L.04.D;

/* **BIT MONITORS** */

/*	The following bits produce indications on the Vital Output Board for testing purposes.	*/

ASSIGN	SYNC.WAIT	TO	OUT.SYNC.WAIT.07;
ASSIGN	SYNC	TO	OUT.SYNC.08;
ASSIGN	HEALTH.WAIT.DELAY	TO	OUT.HEALTH.WAIT.DELAY.09;
ASSIGN	HEALTH.WAIT	TO	OUT.HEALTH.WAIT.10;
ASSIGN	SL.OUT.L.04	TO	OUT.OUT.L.04.12;
ASSIGN	SL.IN.L.04	TO	OUT.IN.L.04.13;
ASSIGN	SL.OUT.RESET	TO	OUT.OUT.RESET.14;
ASSIGN	SL.IN.RESET	TO	OUT.IN.RESET.15;
ASSIGN	COMALT	TO	OUT.COMALT.16;

END LOGIC

END PROGRAM

Example 7

Appendix B

MICROLOK_II PROGRAM;

INTERFACE

LOCAL

BOARD: VO_1_SLOT_J3

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

311H,	312H,	331H,	332H,
333H,	334H,	340H,	361H,
361L,	362H,	362L,	431H,
432H,	460H,	460L,	311RK;

BOARD: VO_2_SLOT_J4

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

501H,	512H,	531H,	532H,
533H,	534H,	540H,	611H,
612H,	01LSR,	02LSR,	03LSR,
04LSR,	312RK,	331RK,	332RK;

BOARD: VO_3_SLOT_J5

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

01NL,	01RL,	02NL,	02RL,
03NL,	03RL,	04NL,	04RL,
05LSR,	06LSR,	07LSR,	08LSR,
09LSR,	10LSR,	333RK,	334RK;

BOARD: VO_4_SLOT_J6

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

05NL,	05RL,	06NL,	06RL,
07NL,	07RL,	08NL,	08RL,
09NL,	09RL,	11LSR,	12LSR,
13LSR,	14LSR,	340RK,	361RK;

BOARD: VO_5_SLOT_J7

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

10NL,	10RL,	11NL,	11RL,
12NL,	12RL,	13NL,	13RL,
14NL,	14RL,	15ESZ,	17ESZ,
20ESZ,	21ESZ,	362RK,	431RK;

BOARD: VO_6_SLOT_J8

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

15TSZ,	17TSZ,	12RWCRZ,	432RK,
460RK,	501RK,	512RK,	531RK,
532RK,	533RK,	534RK,	540RK,
611RK,	612RK,	SPARE,	SYNC.OUT;

BOARD: VI_1_SLOT_J9

ADJUSTABLE ENABLE: 1
TYPE: IN16

INPUT:

NORMAL,	VCOR,	01NWC,	01RWC,
02NWC,	02RWC,	03NWC,	03RWC,
04NWC,	04RWC,	05NWC,	05RWC,
06NWC,	06RWC,	07NWC,	07RWC;

BOARD: VI_2_SLOT_J10

ADJUSTABLE ENABLE: 1
TYPE: IN16

INPUT:

08NWC,	08RWC,	09NWC,	09RWC,
10NWC,	10RWC,	11NWC,	11RWC,
12NWC,	12RWC,	13NWC,	13RWC,
14NWC,	14RWC,	SPARE,	SPARE;

BOARD: VI_3_SLOT_J11

ADJUSTABLE ENABLE: 1
TYPE: IN16

INPUT:

01TPS1,	02TPS1,	03TPS1,	04TPS1,
05TPS1,	06TPS1,	07TPS1,	08TPS1,
09TPS1,	10TPS1,	11TPS1,	12TPS1,
13TPS1,	14TPS1,	SPARE,	SPARE;

BOARD: VI_4_SLOT_J12

ADJUSTABLE ENABLE: 1
TYPE: IN16

INPUT:

15TPS1,	16TPS1,	17TPS1,	20TPS1,
21TPS1,	26TPS1,	35TPS1,	15WSZ,
17WSZ,	20WSZ,	21WSZ,	PO,
20TSZ,	33NWCRZ,	SPARE,	SPARE;

BOARD: VI_5_SLOT_J13

ADJUSTABLE ENABLE: 1
TYPE: IN16

INPUT:

SPARE,	SPARE,	SPARE,	SPARE,
SPARE,	SPARE,	SPARE,	SPARE,
SPARE,	SPARE,	SPARE,	SPARE,
SPARE,	SPARE,	SPARE,	SYNC.IN;

COMM

/* Synchronization Link */

LINK: SYNC_MASTER

ADJUSTABLE	ENABLE:	1
	PROTOCOL:	MICROLOK.MASTER
ADJUSTABLE	POINT.POINT:	1;
ADJUSTABLE	PORT:	1;
ADJUSTABLE	BAUD:	9600;
ADJUSTABLE	STOPBITS:	1;
ADJUSTABLE	PARITY:	NONE;
ADJUSTABLE	KEY.ON.DELAY:	12;
ADJUSTABLE	KEY.OFF.DELAY:	12;
ADJUSTABLE	STALE.DATA.TIMEOUT:	3:SEC;
ADJUSTABLE	POLLING.INTERVAL:	50:MSEC;
ADJUSTABLE	MASTER.TIMEOUT:	100:MSEC;

ADDRESS: 1

ADJUSTABLE ENABLE: 1

OUTPUT:

SL.OUT.HEALTH,	SL.OUT.RESET,	SL.OUT.SYNC,	SL.OUT.540H,
SL.OUT.512H,	SL.OUT.460H,	SL.OUT.460L,	SL.OUT.501H,
SL.OUT.611H,	SL.OUT.612H,	SL.OUT.432H,	SL.OUT.431H,
SL.OUT.312H,	SL.OUT.311H,	SL.OUT.334H,	SL.OUT.333H,
SL.OUT.332H,	SL.OUT.331H,	SL.OUT.361H,	SL.OUT.361L,
SL.OUT.362H,	SL.OUT.362L,	SL.OUT.340H,	SL.OUT.531H,
SL.OUT.532H,	SL.OUT.533H,	SL.OUT.534H,	SL.OUT.540RK,
SL.OUT.512RK,	SL.OUT.460RK,	SL.OUT.501RK,	SL.OUT.611RK,
SL.OUT.612RK,	SL.OUT.432RK,	SL.OUT.431RK,	SL.OUT.312RK,
SL.OUT.311RK,	SL.OUT.334RK,	SL.OUT.333RK,	SL.OUT.332RK,
SL.OUT.331RK,	SL.OUT.361RK,	SL.OUT.340RK,	SL.OUT.531RK,
SL.OUT.362RK,	SL.OUT.532RK,	SL.OUT.533RK,	SL.OUT.534RK,
SL.OUT.01LS,	SPARE,	SPARE,	SPARE,
SL.OUT.02LS,	SL.OUT.03LS,	SL.OUT.04LS,	SL.OUT.05LS,
SL.OUT.06LS,	SL.OUT.07LS,	SL.OUT.08LS,	SL.OUT.09LS,
SL.OUT.10LS,	SL.OUT.11LS,	SL.OUT.12LS,	SL.OUT.13LS,
SL.OUT.14LS,	SL.OUT.01NL,	SL.OUT.02NL,	SL.OUT.03NL,
SL.OUT.04NL,	SL.OUT.05NL,	SL.OUT.06NL,	SL.OUT.07NL,
SL.OUT.08NL,	SL.OUT.09NL,	SL.OUT.10NL,	SL.OUT.11NL,
SL.OUT.12NL,	SL.OUT.13NL,	SL.OUT.14NL,	SL.OUT.01RL;

ADDRESS: 2

ADJUSTABLE ENABLE: 1

OUTPUT:

SL.OUT.02RL,	SL.OUT.03RL,	SL.OUT.04RL,	SL.OUT.05RL,
SL.OUT.06RL,	SL.OUT.07RL,	SL.OUT.08RL,	SL.OUT.09RL,
SL.OUT.10RL,	SL.OUT.11RL,	SL.OUT.12RL,	SL.OUT.13RL,
SL.OUT.14RL,	SL.OUT.01NWCR,	SL.OUT.02NWCR,	SL.OUT.03NWCR,
SL.OUT.04NWCR,	SL.OUT.05NWCR,	SL.OUT.06NWCR,	SL.OUT.07NWCR,
SL.OUT.08NWCR,	SL.OUT.09NWCR,	SL.OUT.10NWCR,	SL.OUT.11NWCR,
SL.OUT.12NWCR,	SL.OUT.13NWCR,	SL.OUT.14NWCR,	SL.OUT.33NWCR,
SL.OUT.01RWCR,	SL.OUT.02RWCR,	SL.OUT.03RWCR,	SL.OUT.04RWCR,
SL.OUT.05RWCR,	SL.OUT.06RWCR,	SL.OUT.07RWCR,	SL.OUT.08RWCR,
SL.OUT.09RWCR,	SL.OUT.10RWCR,	SL.OUT.11RWCR,	SL.OUT.12RWCR,
SL.OUT.13RWCR,	SL.OUT.14RWCR,	SL.OUT.501AS,	SL.OUT.540AS,
SL.OUT.460AS,	SL.OUT.512AS,	SL.OUT.340AS,	SL.OUT.312AS,
SL.OUT.611_612AS,	SL.OUT.531_532_533AS,	SL.OUT.432_534AS,	SL.OUT.331_431AS,
SL.OUT.362AS,	SL.OUT.332_333AS,	SL.OUT.334AS,	SL.OUT.311AS,
SL.OUT.361AS,	SL.OUT.15ES,	SL.OUT.17ES,	SL.OUT.20ES,
SL.OUT.21ES,	SL.OUT.15TS,	SL.OUT.17TS;	

/* Synchronization Link */

LINK: SYNC_SLAVE

ADJUSTABLE	ENABLE:	1
	PROTOCOL:	MICROLOK.SLAVE
ADJUSTABLE	POINT.POINT:	0;
ADJUSTABLE	PORT:	2;
ADJUSTABLE	BAUD:	9600;
ADJUSTABLE	STOPBITS:	1;
ADJUSTABLE	PARITY:	NONE;
ADJUSTABLE	KEY.ON.DELAY:	12;
ADJUSTABLE	KEY.OFF.DELAY:	12;
ADJUSTABLE	STALE.DATA.TIMEOUT:	3:SEC;

ADDRESS: 1

ADJUSTABLE ENABLE: 1

INPUT:

SL.IN.HEALTH,	SL.IN.RESET,	SL.IN.SYNC,	SL.IN.540H,
SL.IN.512H,	SL.IN.460H,	SL.IN.460L,	SL.IN.501H,
SL.IN.611H,	SL.IN.612H,	SL.IN.432H,	SL.IN.431H,
SL.IN.312H,	SL.IN.311H,	SL.IN.334H,	SL.IN.333H,
SL.IN.332H,	SL.IN.331H,	SL.IN.361H,	SL.IN.361L,
SL.IN.362H,	SL.IN.362L,	SL.IN.340H,	SL.IN.531H,

SL.IN.532H, SL.IN.533H, SL.IN.534H, SL.IN.540RK,
SL.IN.512RK, SL.IN.460RK, SL.IN.501RK, SL.IN.611RK,

SL.IN.612RK, SL.IN.432RK, SL.IN.431RK, SL.IN.312RK,
SL.IN.311RK, SL.IN.334RK, SL.IN.333RK, SL.IN.332RK,

SL.IN.331RK, SL.IN.361RK, SL.IN.340RK, SL.IN.531RK,
SL.IN.362RK, SL.IN.532RK, SL.IN.533RK, SL.IN.534RK,

SL.IN.01LS, SPARE, SPARE, SPARE,
SL.IN.02LS, SL.IN.03LS, SL.IN.04LS, SL.IN.05LS,

SL.IN.06LS, SL.IN.07LS, SL.IN.08LS, SL.IN.09LS,
SL.IN.10LS, SL.IN.11LS, SL.IN.12LS, SL.IN.13LS,

SL.IN.14LS, SL.IN.01NL, SL.IN.02NL, SL.IN.03NL,
SL.IN.04NL, SL.IN.05NL, SL.IN.06NL, SL.IN.07NL,

SL.IN.08NL, SL.IN.09NL, SL.IN.10NL, SL.IN.11NL,
SL.IN.12NL, SL.IN.13NL, SL.IN.14NL, SL.IN.01RL;

ADDRESS: 2

ADJUSTABLE ENABLE: 1

INPUT:

SL.IN.02RL, SL.IN.03RL, SL.IN.04RL, SL.IN.05RL,
SL.IN.06RL, SL.IN.07RL, SL.IN.08RL, SL.IN.09RL,

SL.IN.10RL, SL.IN.11RL, SL.IN.12RL, SL.IN.13RL,
SL.IN.14RL, SL.IN.01NWCR, SL.IN.02NWCR, SL.IN.03NWCR,

SL.IN.04NWCR, SL.IN.05NWCR, SL.IN.06NWCR, SL.IN.07NWCR,
SL.IN.08NWCR, SL.IN.09NWCR, SL.IN.10NWCR, SL.IN.11NWCR,

SL.IN.12NWCR, SL.IN.13NWCR, SL.IN.14NWCR, SL.IN.33NWCR,
SL.IN.01RWCR, SL.IN.02RWCR, SL.IN.03RWCR, SL.IN.04RWCR,

SL.IN.05RWCR, SL.IN.06RWCR, SL.IN.07RWCR, SL.IN.08RWCR,
SL.IN.09RWCR, SL.IN.10RWCR, SL.IN.11RWCR, SL.IN.12RWCR,

SL.IN.13RWCR, SL.IN.14RWCR, SL.IN.501AS, SL.IN.540AS,
SL.IN.460AS, SL.IN.512AS, SL.IN.340AS, SL.IN.312AS,

SL.IN.611_612AS, SL.IN.531_532_533AS, SL.IN.432_534AS, SL.IN.331_431AS,
SL.IN.362AS, SL.IN.332_333AS, SL.IN.334AS, SL.IN.311AS,

SL.IN.361AS, SL.IN.15ES, SL.IN.17ES, SL.IN.20ES,
SL.IN.21ES, SL.IN.15TS, SL.IN.17TS;

/* Serial link to Normal Non-Vital Processor */

LINK: NORM_NV_PROC

ADJUSTABLE	ENABLE:	1
	PROTOCOL:	GENISYS.SLAVE
ADJUSTABLE	POINT.POINT:	1;
ADJUSTABLE	PORT:	3;
ADJUSTABLE	BAUD:	9600;
ADJUSTABLE	STOPBITS:	1;
ADJUSTABLE	PARITY:	NONE;
ADJUSTABLE	KEY.ON.DELAY:	12;
ADJUSTABLE	KEY.OFF.DELAY:	12;
ADJUSTABLE	STALE.DATA.TIMEOUT:	3:SEC;

ADDRESS: 1

ADJUSTABLE ENABLE: 1

/* Bits out TO Normal Non-Vital Processor */

NV.OUTPUT:

01NWCR.NNV, 01RWCR.NNV, 02NWCR.NNV, 02RWCR.NNV,
03NWCR.NNV, 03RWCR.NNV, 04NWCR.NNV, 04RWCR.NNV,

05NWCR.NNV, 05RWCR.NNV, 06NWCR.NNV, 06RWCR.NNV,
07NWCR.NNV, 07RWCR.NNV, 08NWCR.NNV, 08RWCR.NNV,

09NWCR.NNV, 09RWCR.NNV, 10NWCR.NNV, 10RWCR.NNV,
11NWCR.NNV, 11RWCR.NNV, 12NWCR.NNV, 12RWCR.NNV,

13NWCR.NNV, 13RWCR.NNV, 14NWCR.NNV, 14RWCR.NNV,
01LS.NNV, 02LS.NNV, 03LS.NNV, 04LS.NNV,

05LS.NNV, 06LS.NNV, 07LS.NNV, 08LS.NNV,
09LS.NNV, 10LS.NNV, 11LS.NNV, 12LS.NNV,

13LS.NNV, 14LS.NNV, 01TPS.NNV, 02TPS.NNV,
03TPS.NNV, 04TPS.NNV, 05TPS.NNV, 06TPS.NNV,

07TPS.NNV, 08TPS.NNV, 09TPS.NNV, 10TPS.NNV,
11TPS.NNV, 12TPS.NNV, 13TPS.NNV, 14TPS.NNV,

15TPS.NNV, 16TPS.NNV, 17TPS.NNV, 20TPS.NNV,
21TPS.NNV, 26TPS.NNV, 35TPS.NNV, 01ES.NNV,

02ES.NNV, 03ES.NNV, 04ES.NNV, 05ES.NNV,
06ES.NNV, 07ES.NNV, 08ES.NNV, 09ES.NNV,

10ES.NNV, 11ES.NNV, 12ES.NNV, 13ES.NNV,
14ES.NNV, 15ES.NNV, 16ES.NNV, 17ES.NNV,

20ES.NNV, 21ES.NNV, 01WS.NNV, 02WS.NNV,
03WS.NNV, 04WS.NNV, 05WS.NNV, 06WS.NNV,

07WS.NNV, 08WS.NNV, 09WS.NNV, 10WS.NNV,
11WS.NNV, 12WS.NNV, 13WS.NNV, 14WS.NNV,

15WS.NNV, 16WS.NNV, 17WS.NNV, 20WS.NNV,
21WS.NNV, 311H.NNV, 312H.NNV, 331H.NNV,

332H.NNV, 333H.NNV, 334H.NNV, 340H.NNV,
361H.NNV, 361L.NNV, 362H.NNV, 362L.NNV,

431H.NNV, 432H.NNV, 460H.NNV, 460L.NNV,
501H.NNV, 512H.NNV, 531H.NNV, 532H.NNV,

533H.NNV, 534H.NNV, 540H.NNV, 611H.NNV,
612H.NNV, SPARE, SPARE, W1.V.HEALTH.NNV,

SL.SYNC.COM.NNV;

/* Bits in FROM Normal Non-Vital Processor */

NV.INPUT:

540R.NNV, 460R.NNV, 501R.NNV, 512R.NNV,
340R.NNV, 611R.NNV, 612R.NNV, 531R.NNV,

532R.NNV, 533R.NNV, 534R.NNV, 432R.NNV,
431R.NNV, 331R.NNV, 332R.NNV, 333R.NNV,

334R.NNV, 311R.NNV, 312R.NNV, 361R.NNV,
362R.NNV, 01NLP.NNV, 01RLP.NNV, 02NLP.NNV,

02RLP.NNV, 03NLP.NNV, 03RLP.NNV, 04NLP.NNV,
04RLP.NNV, 05NLP.NNV, 05RLP.NNV, 06NLP.NNV,

06RLP.NNV, 07NLP.NNV, 07RLP.NNV, 08NLP.NNV,
08RLP.NNV, 09NLP.NNV, 09RLP.NNV, 10NLP.NNV,

10RLP.NNV, 11NLP.NNV, 11RLP.NNV, 12NLP.NNV,
12RLP.NNV, 13NLP.NNV, 13RLP.NNV, 14NLP.NNV,

14RLP.NNV,	SPARE,	SPARE,	SPARE,
SPARE,	SPARE,	SPARE,	SL.N.NV.HEALTH;

/* Serial link to Standby Non-Vital Processor */

LINK: STBY_NV_PROC

ADJUSTABLE	ENABLE:	1
	PROTOCOL:	GENISYS.SLAVE
ADJUSTABLE	POINT.POINT:	1;
ADJUSTABLE	PORT:	4;
ADJUSTABLE	BAUD:	9600;
ADJUSTABLE	STOPBITS:	1;
ADJUSTABLE	PARITY:	NONE;
ADJUSTABLE	KEY.ON.DELAY:	12;
ADJUSTABLE	KEY.OFF.DELAY:	12;
ADJUSTABLE	STALE.DATA.TIMEOUT:	3:SEC;

ADDRESS: 1

ADJUSTABLE ENABLE: 1

/* Bits out TO Standby Non-Vital Processor */

NV.OUTPUT:

01NWCR.SNV,	01RWCR.SNV,	02NWCR.SNV,	02RWCR.SNV,
03NWCR.SNV,	03RWCR.SNV,	04NWCR.SNV,	04RWCR.SNV,
05NWCR.SNV,	05RWCR.SNV,	06NWCR.SNV,	06RWCR.SNV,
07NWCR.SNV,	07RWCR.SNV,	08NWCR.SNV,	08RWCR.SNV,
09NWCR.SNV,	09RWCR.SNV,	10NWCR.SNV,	10RWCR.SNV,
11NWCR.SNV,	11RWCR.SNV,	12NWCR.SNV,	12RWCR.SNV,
13NWCR.SNV,	13RWCR.SNV,	14NWCR.SNV,	14RWCR.SNV,
01LS.SNV,	02LS.SNV,	03LS.SNV,	04LS.SNV,
05LS.SNV,	06LS.SNV,	07LS.SNV,	08LS.SNV,
09LS.SNV,	10LS.SNV,	11LS.SNV,	12LS.SNV,
13LS.SNV,	14LS.SNV,	01TPS.SNV,	02TPS.SNV,
03TPS.SNV,	04TPS.SNV,	05TPS.SNV,	06TPS.SNV,
07TPS.SNV,	08TPS.SNV,	09TPS.SNV,	10TPS.SNV,
11TPS.SNV,	12TPS.SNV,	13TPS.SNV,	14TPS.SNV,

15TPS.SNV, 16TPS.SNV, 17TPS.SNV, 20TPS.SNV,
21TPS.SNV, 26TPS.SNV, 35TPS.SNV, 01ES.SNV,

02ES.SNV, 03ES.SNV, 04ES.SNV, 05ES.SNV,
06ES.SNV, 07ES.SNV, 08ES.SNV, 09ES.SNV,

10ES.SNV, 11ES.SNV, 12ES.SNV, 13ES.SNV,
14ES.SNV, 15ES.SNV, 16ES.SNV, 17ES.SNV,

20ES.SNV, 21ES.SNV, 01WS.SNV, 02WS.SNV,
03WS.SNV, 04WS.SNV, 05WS.SNV, 06WS.SNV,

07WS.SNV, 08WS.SNV, 09WS.SNV, 10WS.SNV,
11WS.SNV, 12WS.SNV, 13WS.SNV, 14WS.SNV,

15WS.SNV, 16WS.SNV, 17WS.SNV, 20WS.SNV,
21WS.SNV, 311H.SNV, 312H.SNV, 331H.SNV,

332H.SNV, 333H.SNV, 334H.SNV, 340H.SNV,
361H.SNV, 361L.SNV, 362H.SNV, 362L.SNV,

431H.SNV, 432H.SNV, 460H.SNV, 460L.SNV,
501H.SNV, 512H.SNV, 531H.SNV, 532H.SNV,

533H.SNV, 534H.SNV, 540H.SNV, 611H.SNV,
612H.SNV, SPARE, SPARE, W1.V.HEALTH.SNV,

SL.SYNC.COM.SNV;

/* Bits in FROM Standby Non-Vital Processor */

NV.INPUT:

540R.SNV, 460R.SNV, 501R.SNV, 512R.SNV,
340R.SNV, 611R.SNV, 612R.SNV, 531R.SNV,

532R.SNV, 533R.SNV, 534R.SNV, 432R.SNV,
431R.SNV, 331R.SNV, 332R.SNV, 333R.SNV,

334R.SNV, 311R.SNV, 312R.SNV, 361R.SNV,
362R.SNV, 01NLP.SNV, 01RLP.SNV, 02NLP.SNV,

02RLP.SNV, 03NLP.SNV, 03RLP.SNV, 04NLP.SNV,
04RLP.SNV, 05NLP.SNV, 05RLP.SNV, 06NLP.SNV,

06RLP.SNV, 07NLP.SNV, 07RLP.SNV, 08NLP.SNV,
08RLP.SNV, 09NLP.SNV, 09RLP.SNV, 10NLP.SNV,

10RLP.SNV, 11NLP.SNV, 11RLP.SNV, 12NLP.SNV,
12RLP.SNV, 13NLP.SNV, 13RLP.SNV, 14NLP.SNV,

14RLP.SNV, SPARE, SPARE, SPARE,
SPARE, SPARE, SPARE, SL.S.NV.HEALTH;

BOOLEAN BITS

/* **HOT STANDBY** */

SYS.RESET,
N.NV.HEALTH, S.NV.HEALTH, N.NV.HEALTH.0, N.NV.HEALTH.1,
S.NV.HEALTH.0, S.NV.HEALTH.1,
SL.IN.HEALTH.0, SL.IN.HEALTH.1, IN.HEALTH,
SYNC, SYNC.WAIT, STAND.ALONE.SYNC.DELAY, STAND.ALONE.SYNC,
DEFAULT.NORMAL.SYNC,
GROUP.01.S_R, GROUP.02.S_R, GROUP.03.S_R, GROUP.04.S_R,
GROUP.05.S_R, GROUP.06.S_R, GROUP.07.S_R,

COMALT,
540H.S_R, 512H.S_R, 460H.S_R, 460L.S_R,
501H.S_R, 611H.S_R, 612H.S_R, 432H.S_R,
431H.S_R, 312H.S_R, 311H.S_R, 334H.S_R,
333H.S_R, 332H.S_R, 331H.S_R, 361H.S_R,
361L.S_R, 362H.S_R, 362L.S_R, 340H.S_R,
531H.S_R, 532H.S_R, 533H.S_R, 534H.S_R,
540RK.S_R, 512RK.S_R, 460RK.S_R, 501RK.S_R,
611RK.S_R, 612RK.S_R, 432RK.S_R, 431RK.S_R,
312RK.S_R, 311RK.S_R, 334RK.S_R, 333RK.S_R,
332RK.S_R, 331RK.S_R, 361RK.S_R, 362RK.S_R,
340RK.S_R, 531RK.S_R, 532RK.S_R, 533RK.S_R,
534RK.S_R,
QUICK.HEALTH.START, QUICK.HEALTH,
01LS.S_R, 02LS.S_R, 03LS.S_R, 04LS.S_R,
05LS.S_R, 06LS.S_R, 07LS.S_R, 08LS.S_R,
09LS.S_R, 10LS.S_R, 11LS.S_R, 12LS.S_R,
13LS.S_R, 14LS.S_R,
01NL.S_R, 02NL.S_R, 03NL.S_R, 04NL.S_R,

05NL.S_R, 06NL.S_R, 07NL.S_R, 08NL.S_R,
09NL.S_R, 10NL.S_R, 11NL.S_R, 12NL.S_R,
13NL.S_R, 14NL.S_R,
01RL.S_R, 02RL.S_R, 03RL.S_R, 04RL.S_R,
05RL.S_R, 06RL.S_R, 07RL.S_R, 08RL.S_R,
09RL.S_R, 10RL.S_R, 11RL.S_R, 12RL.S_R,
13RL.S_R, 14RL.S_R,
01NWCR.S_R, 02NWCR.S_R, 03NWCR.S_R, 04NWCR.S_R,
05NWCR.S_R, 06NWCR.S_R, 07NWCR.S_R, 08NWCR.S_R,
09NWCR.S_R, 10NWCR.S_R, 11NWCR.S_R, 12NWCR.S_R,
13NWCR.S_R, 14NWCR.S_R, 33NWCR.S_R,
01RWCR.S_R, 02RWCR.S_R, 03RWCR.S_R, 04RWCR.S_R,
05RWCR.S_R, 06RWCR.S_R, 07RWCR.S_R, 08RWCR.S_R,
09RWCR.S_R, 10RWCR.S_R, 11RWCR.S_R, 12RWCR.S_R,
13RWCR.S_R, 14RWCR.S_R,
501AS.S_R, 540AS.S_R, 460AS.S_R, 512AS.S_R,
340AS.S_R, 312AS.S_R, 611_612AS.S_R, 531_532_533AS.S_R,
432_534AS.S_R, 331_431AS.S_R, 362AS.S_R, 332_333AS.S_R,
334AS.S_R, 311AS.S_R, 361AS.S_R,
15ES.S_R, 17ES.S_R, 20ES.S_R, 21ES.S_R,
15TS.S_R, 17TS.S_R,

/* **ZONE SPECIFIC** */

FLASH, POPS, 01NWZ, 01RWZ,
02NWZ, 02RWZ, 03NWZ, 03RWZ,
04NWZ, 04RWZ, 05NWZ, 05RWZ,
06NWZ, 06RWZ, 07NWZ, 07RWZ,
08NWZ, 08RWZ, 09NWZ, 09RWZ,
10NWZ, 10RWZ, 11NWZ, 11RWZ,
12NWZ, 12RWZ, 13NWZ, 13RWZ,
14NWZ, 14RWZ, 33NWCR, 501AS,
540AS, 460AS, 512AS, 340AS,
312AS, 611_612AS, 531_532_533AS, 432_534AS,
331_431AS, 362AS, 332_333AS, 334AS,
311AS, 361AS, 501_540TE, 460TE,
512TE, 312_340TE, 611_612TE, 432_531_534TE,
331_431_362TE, 332_334TE, 01ES, 02ES,
03ES, 04ES, 05ES, 06ES,
07ES, 08ES, 09ES, 10ES,
11ES, 12ES, 13ES, 14ES,
15ES, 16ES, 17ES, 20ES,
21ES, 01WS, 02WS, 03WS,

04WS, 05WS, 06WS, 07WS,
08WS, 09WS, 10WS, 11WS,
12WS, 13WS, 14WS, 15WS,
16WS, 17WS, 20WS, 21WS,
01TPS, 02TPS, 03TPS, 04TPS,
05TPS, 06TPS, 07TPS, 08TPS,
09TPS, 10TPS, 11TPS, 12TPS,
13TPS, 14TPS, 15TPS, 16TPS,
17TPS, 20TPS, 21TPS, 26TPS,
35TPS, 26TE, 35TE, 311_361TE,
01TE, 02TE, 03TE, 04TE,
05TE, 06TE, 07TE, 08TE,
09TE, 10TE, 11TE, 12TE,
13TE, 14TE, 15TE, 16TE,
17TE, 20TE, 21TE, 01NWCR,
01RWCR, 02NWCR, 02RWCR, 03NWCR,
03RWCR, 04NWCR, 04RWCR, 05NWCR,
05RWCR, 06NWCR, 06RWCR, 07NWCR,
07RWCR, 08NWCR, 08RWCR, 09NWCR,
09RWCR, 10NWCR, 10RWCR, 11NWCR,
11RWCR, 12NWCR, 12RWCR, 13NWCR,
13RWCR, 14NWCR, 14RWCR, 17TS,
15TS, 20TS, 531H1, 532H1,
533H1, 534H1, 460H1, 432H1,
01LS, 02LS, 03LS, 04LS,
05LS, 06LS, 07LS, 08LS,
09LS, 10LS, 11LS, 12LS,
13LS, 14LS, 311R, 312R,
331R, 332R, 333R, 334R,
340R, 361R, 362R, 431R,
432R, 460R, 501R, 512R,
531R, 532R, 533R, 534R,
540R, 611R, 612R, 01NLP,
01RLP, 02NLP, 02RLP, 03NLP,
03RLP, 04NLP, 04RLP, 05NLP,
05RLP, 06NLP, 06RLP, 07NLP,
07RLP, 08NLP, 08RLP, 09NLP,
09RLP, 10NLP, 10RLP, 11NLP,
11RLP, 12NLP, 12RLP, 13NLP,
13RLP, 14NLP, 14RLP;

TIMER BITS

/* HOT STANDBY */

/* SL.OUT.RESET is sent from the Normal unit to the Standby unit when the Normal unit determines there is a disagreement in bit states. It is delayed to allow the Standby unit time to synchronize. The exact setting for this bit is based on the needs of each application. It should be as short as possible without effecting reliability. */

ADJUSTABLE	SL.OUT.RESET:	SET = 2:SEC	CLEAR = 0:SEC;

/* SYS.RESET is an internal bit that RESETS the Standby unit if it is out of synchronization with the online Normal unit. It is slightly delayed to insure that the Standby unit does not falsely reset. The exact setting for this bit is based on the needs of each application. It should be as short as possible without effecting reliability. */

ADJUSTABLE	SYS.RESET:	SET = 2:SEC	CLEAR = 0:SEC;

/* The GROUP.RESET bits represent groups of individual bit resets. They are slightly delayed to insure that the Standby unit does not reset falsely. GROUP.03.V.RESET contains bits that are "more vital" such as Switch Locking or Route Locking, therefore they are given a shorter reset time. The exact setting for these bits is based on the needs of each application. They should be as short as possible without effecting reliability. */

ADJUSTABLE	GROUP.01.S_R:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	GROUP.02.S_R:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	GROUP.03.S_R:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	GROUP.04.S_R:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	GROUP.05.S_R:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	GROUP.06.S_R:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	GROUP.07.S_R:	SET = 0:SEC	CLEAR = 1:SEC;

/* STAND.ALONE.SYNC.DELAY is a slow set bit that allows the unit to stabilize before VCOR is referenced for SYNC. */

ADJUSTABLE	STAND.ALONE.SY	SET = 1:SEC	CLEAR = 0:SEC;

/* SYNC.WAIT is a slow set internal bit that allows serial communication to stabilize after the unit is powered up before synchronization is verified. It should always be set for 5 seconds or longer. */

ADJUSTABLE	SYNC.WAIT:	SET = 5:SEC	CLEAR = 0:SEC;

/* ZONE SPECIFIC */

ADJUSTABLE	FLASH:	SET = 500:MSEC	CLEAR =
ADJUSTABLE	DEFAULT.NORMA	SET = 30:SEC	CLEAR = 0:SEC;

ADJUSTABLE	W1.V.HEALTH.NN	SET = 1:SEC	CLEAR = 1:SEC;
ADJUSTABLE	W1.V.HEALTH.SNV	SET = 1:SEC	CLEAR = 1:SEC;
ADJUSTABLE	SL.SYNC.COM.SNV	SET = 1:SEC	CLEAR = 1:SEC;
ADJUSTABLE	SL.SYNC.COM.NN	SET = 1:SEC	CLEAR = 1:SEC;
ADJUSTABLE	SL.OUT.HEALTH:	SET = 1:SEC	CLEAR = 1:SEC;
ADJUSTABLE	SL.IN.HEALTH.0:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	SL.IN.HEALTH.1:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	IN.HEALTH:	SET = 4:SEC	CLEAR = 1:SEC;
ADJUSTABLE	N.NV.HEALTH.0:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	N.NV.HEALTH.1:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	N.NV.HEALTH:	SET = 4:SEC	CLEAR = 1:SEC;
ADJUSTABLE	S.NV.HEALTH.0:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	S.NV.HEALTH.1:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	S.NV.HEALTH:	SET = 4:SEC	CLEAR = 1:SEC;
ADJUSTABLE	QUICK.HEALTH.ST	SET = 0:SEC	CLEAR = 20:SEC;
ADJUSTABLE	QUICK.HEALTH:	SET = 0:SEC	CLEAR = 1:SEC;
ADJUSTABLE	501_540TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	460TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	512TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	312_340TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	611_612TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	432_531_534TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	331_431_362TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	332_334TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	311_361TE:	SET = 14:SEC	CLEAR = 0:SEC;
ADJUSTABLE	01TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	02TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	03TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	04TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	05TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	06TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	07TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	08TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	09TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	10TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	11TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	12TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	13TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	14TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	15TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	16TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	17TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	20TE:	SET = 5:SEC	CLEAR = 0:SEC;

ADJUSTABLE	21TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	26TE:	SET = 5:SEC	CLEAR = 0:SEC;
ADJUSTABLE	35TE:	SET = 5:SEC	CLEAR = 0:SEC;

CONSTANTS BOOLEAN

ONE	=	1;
ZERO	=	0;

CONFIGURATION

SYSTEM

ADJUSTABLE	DEBUG_PORT_ADDRESS:	1;
ADJUSTABLE	DEBUG_PORT_BAUDRATE:	9600;
ADJUSTABLE	LOGIC_TIMEOUT:	2:SEC;
ADJUSTABLE	DELAY_RESET:	3:SEC;

LOGIC BEGIN

/* **SYSTEM BITS** */

ASSIGN	ONE	TO	CPS.ENABLE;
ASSIGN	(ONE + PO) * (SL.OUT.501AS * SL.OUT.540AS * SL.OUT.460AS * SL.OUT.512AS * SL.OUT.340AS * SL.OUT.312AS * SL.OUT.611_612AS * SL.OUT.531_532_533AS * SL.OUT.432_534AS * SL.OUT.331_431AS * SL.OUT.362AS * SL.OUT.332_333AS * SL.OUT.334AS * SL.OUT.311AS * SL.OUT.361AS + POPS)	TO	POPS;
ASSIGN	~FLASH	TO	FLASH;

/* **HOT STANDBY SYSTEM BITS** */

ASSIGN	SYS.RESET	TO	RESET;
ASSIGN	SYNC * ~SL.SYNC.COM.NNV	TO	SL.SYNC.COM.NNV;
ASSIGN	SYNC * ~SL.SYNC.COM.SNV	TO	SL.SYNC.COM.SNV;

/* **HOT STANDBY RESET BITS** */

SYS.RESET is a slow set bit that only functions in the Standby unit.
The bits in the assign statement function as follows:

- ~NORMAL insures that only the Standby unit can be RESET.
- VCOR insures that the unit will only RESET if the Normal unit is online.
- SYNC insures that the unit will only RESET if it is producing outputs. This prevents the unit from continually resetting and gives it an opportunity to achieve synchronization.
- SL.IN.RESET comes from the Normal unit and forces the Standby unit to RESET.
- ~SL.IN.HEALTH insures the Standby unit will RESET itself if serial communication is lost between the units.
- ~HEALTH.WAIT insures the Standby unit will not RESET itself before serial communication is established when the Normal unit is coming online.
- SL.IN.SYNC insures that the Standby unit will only RESET itself if the Normal unit is in sync.
- ~SYNC insures that the Standby unit will RESET itself if both units are powered up simultaneously and do not achieve synchronization. This permits the Normal unit to take control.
- SYNC.WAIT delays the RESET until the Standby unit has an opportunity to verify synchronization with the Normal unit.
- GROUP.01.RESET, GROUP.02.RESET, and GROUP.03.V.RESET are groups of individual reset bits.

```
ASSIGN ~NORMAL * VCOR * SYNC * SYNC.IN *
       (SL.IN.RESET + (~IN.HEALTH * SL.IN.SYNC *
       ~QUICK.HEALTH) +
       ~GROUP.01.S_R + ~GROUP.02.S_R +
       ~GROUP.03.S_R + ~GROUP.04.S_R +
       ~GROUP.05.S_R + ~GROUP.06.S_R +
       ~GROUP.07.S_R)                    TO  SYS.RESET;
```

/* SL.OUT.RESET is a slow set bit that is sent from the Normal unit to the Standby unit when any output bit is out of sync. It is primarily controlled by the GROUP.RESET bits, however, the SYNC bit is also required so that the Normal unit cannot reset the Standby unit if the Normal is being powered up and cannot achieve synchronization with the Standby. */

```
ASSIGN (NORMAL * SYNC) * ~IN.HEALTH +
       (~GROUP.01.S_R + ~GROUP.02.S_R +
       ~GROUP.03.S_R + ~GROUP.04.S_R +
       ~GROUP.05.S_R + ~GROUP.06.S_R +
       ~GROUP.07.S_R)                    TO  SL.OUT.RESET;
```

/* GROUP.01.RESET, GROUP.02.RESET, and GROUP.03.V.RESET are groups of individual reset bits (though only one RESET bit is assigned to each for testing). The individual reset bits are grouped together to simplify the SYS.RESET equation and to allow for a longer time delay for non-synchronous situations which may be caused by serial communication delays. Three groups are used for testing but the maximum number of groups is unlimited. Multiple groups should be used to limit the number of bits so that continuous changes of bit states will not by misinterpreted as a non-synchronous condition. GROUP.03.V.RESET represents groups of bits that are "more vital" such as Switch Locks and Route Locks. This group is given the absolutely shortest time delay possible while still maintaining reliability. */

/* **HOT STANDBY SYNCHRONIZATION BITS** */

/*	SYNC suppresses all outputs of the unit being brought online until they are verified to be synchronous with the unit currently in control or the other unit's VCOR is down. Once it is set it is stuck high until the unit is powered down. SL.OUT.SYNC is sent out to the other unit. It is utilized by the Standby unit in the SYS.RESET assign statement. */

ASSIGN	SYNC + STAND.ALONE.SYNC + (GROUP.01.S_R * GROUP.02.S_R * GROUP.03.S_R * GROUP.04.S_R * GROUP.05.S_R * GROUP.06.S_R * GROUP.07.S_R)	TO	SYNC, SL.OUT.SYNC, SYNC.OUT, LED.1;

ASSIGN	540H.S_R * 512H.S_R * 460H.S_R * 460L.S_R * 501H.S_R * 611H.S_R * 612H.S_R * 432H.S_R * 431H.S_R * 312H.S_R * 311H.S_R * 334H.S_R * 333H.S_R * 332H.S_R * 331H.S_R * 361H.S_R * 361L.S_R * 362H.S_R * 362L.S_R * 340H.S_R * 531H.S_R * 532H.S_R * 533H.S_R * 534H.S_R	TO	GROUP.01.S_R;
ASSIGN	01LS.S_R * 02LS.S_R * 03LS.S_R * 04LS.S_R * 05LS.S_R * 06LS.S_R * 07LS.S_R * 08LS.S_R * 09LS.S_R * 10LS.S_R * 11LS.S_R * 12LS.S_R * 13LS.S_R * 14LS.S_R	TO	GROUP.02.S_R;
ASSIGN	01NL.S_R * 02NL.S_R * 03NL.S_R * 04NL.S_R * 05NL.S_R * 06NL.S_R * 07NL.S_R * 08NL.S_R * 09NL.S_R * 10NL.S_R * 11NL.S_R * 12NL.S_R * 13NL.S_R * 14NL.S_R * 01RL.S_R * 02RL.S_R * 03RL.S_R * 04RL.S_R * 05RL.S_R * 06RL.S_R * 07RL.S_R * 08RL.S_R * 09RL.S_R * 10RL.S_R * 11RL.S_R * 12RL.S_R * 13RL.S_R * 14RL.S_R	TO	GROUP.03.S_R;
ASSIGN	01NWCR.S_R * 02NWCR.S_R * 03NWCR.S_R * 04NWCR.S_R * 05NWCR.S_R * 06NWCR.S_R * 07NWCR.S_R * 08NWCR.S_R * 09NWCR.S_R * 10NWCR.S_R * 11NWCR.S_R * 12NWCR.S_R * 13NWCR.S_R * 14NWCR.S_R * 33NWCR.S_R * 01RWCR.S_R * 02RWCR.S_R * 03RWCR.S_R * 04RWCR.S_R * 05RWCR.S_R * 06RWCR.S_R * 07RWCR.S_R * 08RWCR.S_R * 09RWCR.S_R * 10RWCR.S_R * 11RWCR.S_R * 12RWCR.S_R * 13RWCR.S_R * 14RWCR.S_R	TO	GROUP.04.S_R;
ASSIGN	501AS.S_R * 540AS.S_R * 460AS.S_R * 512AS.S_R * 340AS.S_R * 312AS.S_R * 611_612AS.S_R * 531_532_533AS.S_R * 432_534AS.S_R * 331_431AS.S_R * 362AS.S_R * 332_333AS.S_R * 334AS.S_R * 311AS.S_R * 361AS.S_R	TO	GROUP.05.S_R;
ASSIGN	15ES.S_R * 17ES.S_R * 20ES.S_R * 21ES.S_R * 15TS.S_R * 17TS.S_R	TO	GROUP.06.S_R;

ASSIGN	540RK.S_R + 512RK.S_R + 460RK.S_R + 501RK.S_R + 611RK.S_R + 612RK.S_R + 432RK.S_R + 431RK.S_R + 312RK.S_R + 311RK.S_R + 334RK.S_R + 333RK.S_R + 332RK.S_R + 331RK.S_R + 361RK.S_R + 362RK.S_R + 340RK.S_R + 531RK.S_R + 532RK.S_R + 533RK.S_R + 534RK.S_R	TO	GROUP.07.S_R;

/* SYNC.WAIT is a slow set bit that suppresses verification of bits in the unit being brought online until it is powered up and both the unit and the serial communication link are stable. Once it is set it is stuck high until the unit is powered down. */

ASSIGN	SYNC.WAIT + (VCOR * IN.HEALTH)	TO	SYNC.WAIT;

/* STAND.ALONE.SYNC.DELAY is a slow set bit that sets one second after the unit is powered up and begins processing. Its purpose is to allow the unit to stabalize before the unit attempts to achieve STAND.ALONE.SYNC. */

ASSIGN	ONE	TO	STAND.ALONE.SYNC.DELAY;

/* STAND.ALONE.SYNC is an internal bit that will set the SYNC bit one second after the unit is powered up if the other unit's VCOR is down. */

ASSIGN	~VCOR * STAND.ALONE.SYNC.DELAY * ~SYNC.IN * ~SL.IN.SYNC + DEFAULT.NORMAL.SYNC	TO	STAND.ALONE.SYNC;

/* DEFAULT.NORMAL.SYNC is a slow set bit that lets the Normal unit achieve SYNC if both units are powered on simultaneously and there is a disagreement between the units. */

ASSIGN	VCOR * STAND.ALONE.SYNC.DELAY * ~SYNC.IN * ~SL.IN.SYNC * NORMAL * IN.HEALTH	TO	DEFAULT.NORMAL.SYNC;

/* HOT STANDBY HEALTH BITS */

ASSIGN	~SL.OUT.HEALTH	TO	SL.OUT.HEALTH;
ASSIGN	SL.IN.HEALTH * ~IN.HEALTH * ~QUICK.HEALTH	TO	QUICK.HEALTH.START;
ASSIGN	QUICK.HEALTH.START * ~IN.HEALTH	TO	QUICK.HEALTH;
NV.ASSIGN	~W1.V.HEALTH.NNV	TO	W1.V.HEALTH.NNV;
NV.ASSIGN	~W1.V.HEALTH.SNV	TO	W1.V.HEALTH.SNV;
ASSIGN	SL.IN.HEALTH	TO	SL.IN.HEALTH.1;
ASSIGN	~SL.IN.HEALTH	TO	SL.IN.HEALTH.0;

ASSIGN	SL.IN.HEALTH.0 * SL.IN.HEALTH.1	TO	IN.HEALTH;
ASSIGN	SL.S.NV.HEALTH	TO	S.NV.HEALTH.1;
ASSIGN	~SL.S.NV.HEALTH	TO	S.NV.HEALTH.0;
ASSIGN	S.NV.HEALTH.0 * S.NV.HEALTH.1	TO	S.NV.HEALTH;
ASSIGN	SL.N.NV.HEALTH	TO	N.NV.HEALTH.1;
ASSIGN	~SL.N.NV.HEALTH	TO	N.NV.HEALTH.0;
ASSIGN	N.NV.HEALTH.0 * N.NV.HEALTH.1	TO	N.NV.HEALTH;

/* **SWITCH MACHINE CONTROL AND CORRESPONDENCE** */

ASSIGN	~SL.OUT.01NWCR * ~SL.OUT.01RWCR * 01NWZ * SL.OUT.01LS	TO	SL.OUT.01NL;
ASSIGN	(01NL.S_R * ~SYNC) + (((SL.OUT.01NL * SL.IN.01NL) + (~SL.OUT.01NL * ~SL.IN.01NL)) * VCOR * SYNC.WAIT)	TO	01NL.S_R;
ASSIGN	SL.OUT.01NL * SYNC * (SL.IN.01NL + ~SL.IN.SYNC + ~VCOR)	TO	01NL;
ASSIGN	~SL.OUT.01NWCR * ~SL.OUT.01RWCR *01RWZ * SL.OUT.01LS	TO	SL.OUT.01RL;
ASSIGN	(01RL.S_R * ~SYNC) + (((SL.OUT.01RL * SL.IN.01RL) + (~SL.OUT.01RL * ~SL.IN.01RL)) * VCOR * SYNC.WAIT)	TO	01RL.S_R;
ASSIGN	SL.OUT.01RL * SYNC * (SL.IN.01RL + ~SL.IN.SYNC + ~VCOR)	TO	01RL;
ASSIGN	(~01RLP * 01NLP * SL.OUT.01LS) + (~01RWZ * 01NWZ) + (~01NWZ * ~01RWZ * 01NWC * (~SYNC + ~VCOR + SL.IN.01NWCR))	TO	01NWZ;
ASSIGN	(~01NLP * 01RLP * SL.OUT.01LS) + (~01NWZ * 01RWZ) + (~01NWZ * ~01RWZ * 01RWC * (~SYNC + ~VCOR + SL.IN.01RWCR))	TO	01RWZ;
ASSIGN	01NWC * ~01RWZ * 01NWZ	TO	SL.OUT.01NWCR;
ASSIGN	(01NWCR.S_R * ~SYNC) + (((SL.OUT.01NWCR * SL.IN.01NWCR) + (~SL.OUT.01NWCR * ~SL.IN.01NWCR)) * VCOR * SYNC.WAIT)	TO	01NWCR.S_R;

ASSIGN	SL.OUT.01NWCR * SYNC	TO	01NWCR;
ASSIGN	01RWC * ~01NWZ * 01RWZ	TO	SL.OUT.01RWCR;
ASSIGN	(01RWCR.S_R * ~SYNC) + (((SL.OUT.01RWCR * SL.IN.01RWCR) + (~SL.OUT.01RWCR * ~SL.IN.01RWCR)) * VCOR * SYNC.WAIT)	TO	01RWCR.S_R;
ASSIGN	SL.OUT.01RWCR * SYNC	TO	01RWCR;
ASSIGN	~SL.OUT.02NWCR * ~SL.OUT.02RWCR * 02NWZ * SL.OUT.02LS	TO	SL.OUT.02NL;
ASSIGN	(02NL.S_R * ~SYNC) + (((SL.OUT.02NL * SL.IN.02NL) + (~SL.OUT.02NL * ~SL.IN.02NL)) * VCOR * SYNC.WAIT)	TO	02NL.S_R;
ASSIGN	SL.OUT.02NL * SYNC * (SL.IN.02NL + ~SL.IN.SYNC + ~VCOR)	TO	02NL;
ASSIGN	~SL.OUT.02NWCR * ~SL.OUT.02RWCR * 02RWZ * SL.OUT.02LS	TO	SL.OUT.02RL;
ASSIGN	(02RL.S_R * ~SYNC) + (((SL.OUT.02RL * SL.IN.02RL) + (~SL.OUT.02RL * ~SL.IN.02RL)) * VCOR * SYNC.WAIT)	TO	02RL.S_R;
ASSIGN	SL.OUT.02RL * SYNC * (SL.IN.02RL + ~SL.IN.SYNC + ~VCOR)	TO	02RL;
ASSIGN	(~02RLP * 02NLP * SL.OUT.02LS) + (~02RWZ * 02NWZ) + (~02NWZ * ~02RWZ * 02NWC * (~SYNC + ~VCOR + SL.IN.02NWCR))	TO	02NWZ;
ASSIGN	(~02NLP * 02RLP * SL.OUT.02LS) + (~02NWZ * 02RWZ) + (~02NWZ * ~02RWZ * 02RWC * (~SYNC + ~VCOR + SL.IN.02RWCR))	TO	02RWZ;
ASSIGN	02NWC * ~02RWZ * 02NWZ	TO	SL.OUT.02NWCR;
ASSIGN	(02NWCR.S_R * ~SYNC) + (((SL.OUT.02NWCR * SL.IN.02NWCR) + (~SL.OUT.02NWCR * ~SL.IN.02NWCR)) * VCOR * SYNC.WAIT)	TO	02NWCR.S_R;
ASSIGN	SL.OUT.02NWCR * SYNC	TO	02NWCR;
ASSIGN	02RWC * ~02NWZ * 02RWZ	TO	SL.OUT.02RWCR;
ASSIGN	(02RWCR.S_R * ~SYNC) + (((SL.OUT.02RWCR * SL.IN.02RWCR) + (~SL.OUT.02RWCR * ~SL.IN.02RWCR)) * VCOR * SYNC.WAIT)	TO	02RWCR.S_R;

ASSIGN	SL.OUT.02RWCR * SYNC	TO	02RWCR;
ASSIGN	~SL.OUT.03NWCR * ~SL.OUT.03RWCR * 03NWZ * SL.OUT.03LS	TO	SL.OUT.03NL;
ASSIGN	(03NL.S_R * ~SYNC) + (((SL.OUT.03NL * SL.IN.03NL) + (~SL.OUT.03NL * ~SL.IN.03NL)) * VCOR * SYNC.WAIT)	TO	03NL.S_R;
ASSIGN	SL.OUT.03NL * SYNC * (SL.IN.03NL + ~SL.IN.SYNC + ~VCOR)	TO	03NL;
ASSIGN	~SL.OUT.03NWCR * ~SL.OUT.03RWCR * 03RWZ * SL.OUT.03LS	TO	SL.OUT.03RL;
ASSIGN	(03RL.S_R * ~SYNC) + (((SL.OUT.03RL * SL.IN.03RL) + (~SL.OUT.03RL * ~SL.IN.03RL)) * VCOR * SYNC.WAIT)	TO	03RL.S_R;
ASSIGN	SL.OUT.03RL * SYNC * (SL.IN.03RL + ~SL.IN.SYNC + ~VCOR)	TO	03RL;
ASSIGN	(~03RLP * 03NLP * SL.OUT.03LS) + (~03RWZ * 03NWZ) + (~03NWZ * ~03RWZ * 03NWC * (~SYNC + ~VCOR + SL.IN.03NWCR))	TO	03NWZ;
ASSIGN	(~03NLP * 03RLP * SL.OUT.03LS) + (~03NWZ * 03RWZ) + (~03NWZ * ~03RWZ * 03RWC * (~SYNC + ~VCOR + SL.IN.03RWCR))	TO	03RWZ;
ASSIGN	03NWC * ~03RWZ * 03NWZ	TO	SL.OUT.03NWCR;
ASSIGN	(03NWCR.S_R * ~SYNC) + (((SL.OUT.03NWCR * SL.IN.03NWCR) + (~SL.OUT.03NWCR * ~SL.IN.03NWCR)) * VCOR * SYNC.WAIT)	TO	03NWCR.S_R;
ASSIGN	SL.OUT.03NWCR * SYNC	TO	03NWCR;
ASSIGN	03RWC * ~03NWZ * 03RWZ	TO	SL.OUT.03RWCR;
ASSIGN	(03RWCR.S_R * ~SYNC) + (((SL.OUT.03RWCR * SL.IN.03RWCR) + (~SL.OUT.03RWCR * ~SL.IN.03RWCR)) * VCOR * SYNC.WAIT)	TO	03RWCR.S_R;
ASSIGN	SL.OUT.03RWCR * SYNC	TO	03RWCR;
ASSIGN	~SL.OUT.04NWCR * ~SL.OUT.04RWCR * 04NWZ * SL.OUT.04LS	TO	SL.OUT.04NL;
ASSIGN	(04NL.S_R * ~SYNC) + (((SL.OUT.04NL * SL.IN.04NL) + (~SL.OUT.04NL * ~SL.IN.04NL)) * VCOR * SYNC.WAIT)	TO	04NL.S_R;

ASSIGN	SL.OUT.04NL * SYNC * (SL.IN.04NL + ~SL.IN.SYNC + ~VCOR)	TO	04NL;
ASSIGN	~SL.OUT.04NWCR * ~SL.OUT.04RWCR * 04RWZ * SL.OUT.04LS	TO	SL.OUT.04RL;
ASSIGN	(04RL.S_R * ~SYNC) + (((SL.OUT.04RL * SL.IN.04RL) + (~SL.OUT.04RL * ~SL.IN.04RL)) * VCOR * SYNC.WAIT)	TO	04RL.S_R;
ASSIGN	SL.OUT.04RL * SYNC * (SL.IN.04RL + ~SL.IN.SYNC + ~VCOR)	TO	04RL;
ASSIGN	(~04RLP * 04NLP * SL.OUT.04LS) + (~04RWZ * 04NWZ) + (~04NWZ * ~04RWZ * 04NWC * (~SYNC + ~VCOR + SL.IN.04NWCR))	TO	04NWZ;
ASSIGN	(~04NLP * 04RLP * SL.OUT.04LS) + (~04NWZ * 04RWZ) + (~04NWZ * ~04RWZ * 04RWC * (~SYNC + ~VCOR + SL.IN.04RWCR))	TO	04RWZ;
ASSIGN	04NWC * ~04RWZ * 04NWZ	TO	SL.OUT.04NWCR;
ASSIGN	(04NWCR.S_R * ~SYNC) + (((SL.OUT.04NWCR * SL.IN.04NWCR) + (~SL.OUT.04NWCR * ~SL.IN.04NWCR)) * VCOR * SYNC.WAIT)	TO	04NWCR.S_R;
ASSIGN	SL.OUT.04NWCR * SYNC	TO	04NWCR;
ASSIGN	04RWC * ~04NWZ * 04RWZ	TO	SL.OUT.04RWCR;
ASSIGN	(04RWCR.S_R * ~SYNC) + (((SL.OUT.04RWCR * SL.IN.04RWCR) + (~SL.OUT.04RWCR * ~SL.IN.04RWCR)) * VCOR * SYNC.WAIT)	TO	04RWCR.S_R;
ASSIGN	SL.OUT.04RWCR * SYNC	TO	04RWCR;
ASSIGN	~SL.OUT.05NWCR * ~SL.OUT.05RWCR * 05NWZ * SL.OUT.05LS	TO	SL.OUT.05NL;
ASSIGN	(05NL.S_R * ~SYNC) + (((SL.OUT.05NL * SL.IN.05NL) + (~SL.OUT.05NL * ~SL.IN.05NL)) * VCOR * SYNC.WAIT)	TO	05NL.S_R;
ASSIGN	SL.OUT.05NL * SYNC * (SL.IN.05NL + ~SL.IN.SYNC + ~VCOR)	TO	05NL;
ASSIGN	~SL.OUT.05NWCR * ~SL.OUT.05RWCR * 05RWZ * SL.OUT.05LS	TO	SL.OUT.05RL;
ASSIGN	(05RL.S_R * ~SYNC) + (((SL.OUT.05RL * SL.IN.05RL) + (~SL.OUT.05RL * ~SL.IN.05RL)) * VCOR * SYNC.WAIT)	TO	05RL.S_R;

ASSIGN	SL.OUT.05RL * SYNC * (SL.IN.05RL + ~SL.IN.SYNC + ~VCOR)	TO	05RL;
ASSIGN	(~05RLP * 05NLP * SL.OUT.05LS) + (~05RWZ * 05NWZ) + (~05NWZ * ~05RWZ * 05NWC * (~SYNC + ~VCOR + SL.IN.05NWCR))	TO	05NWZ;
ASSIGN	(~05NLP * 05RLP * SL.OUT.05LS) + (~05NWZ * 05RWZ) + (~05NWZ * ~05RWZ * 05RWC * (~SYNC + ~VCOR + SL.IN.05RWCR))	TO	05RWZ;
ASSIGN	05NWC * ~05RWZ * 05NWZ	TO	SL.OUT.05NWCR;
ASSIGN	(05NWCR.S_R * ~SYNC) + (((SL.OUT.05NWCR * SL.IN.05NWCR) + (~SL.OUT.05NWCR * ~SL.IN.05NWCR)) * VCOR * SYNC.WAIT)	TO	05NWCR.S_R;
ASSIGN	SL.OUT.05NWCR * SYNC	TO	05NWCR;
ASSIGN	05RWC * ~05NWZ * 05RWZ	TO	SL.OUT.05RWCR;
ASSIGN	(05RWCR.S_R * ~SYNC) + (((SL.OUT.05RWCR * SL.IN.05RWCR) + (~SL.OUT.05RWCR * ~SL.IN.05RWCR)) * VCOR * SYNC.WAIT)	TO	05RWCR.S_R;
ASSIGN	SL.OUT.05RWCR * SYNC	TO	05RWCR;
ASSIGN	~SL.OUT.06NWCR * ~SL.OUT.06RWCR * 06NWZ * SL.OUT.06LS	TO	SL.OUT.06NL;
ASSIGN	(06NL.S_R * ~SYNC) + (((SL.OUT.06NL * SL.IN.06NL) + (~SL.OUT.06NL * ~SL.IN.06NL)) * VCOR * SYNC.WAIT)	TO	06NL.S_R;
ASSIGN	SL.OUT.06NL * SYNC * (SL.IN.06NL + ~SL.IN.SYNC + ~VCOR)	TO	06NL;
ASSIGN	~SL.OUT.06NWCR * ~SL.OUT.06RWCR * 06RWZ * SL.OUT.06LS	TO	SL.OUT.06RL;
ASSIGN	(06RL.S_R * ~SYNC) + (((SL.OUT.06RL * SL.IN.06RL) + (~SL.OUT.06RL * ~SL.IN.06RL)) * VCOR * SYNC.WAIT)	TO	06RL.S_R;
ASSIGN	SL.OUT.06RL * SYNC * (SL.IN.06RL + ~SL.IN.SYNC + ~VCOR)	TO	06RL;
ASSIGN	(~06RLP * 06NLP * SL.OUT.06LS) + (~06RWZ * 06NWZ) + (~06NWZ * ~06RWZ * 06NWC * (~SYNC + ~VCOR + SL.IN.06NWCR))	TO	06NWZ;
ASSIGN	(~06NLP * 06RLP * SL.OUT.06LS) + (~06NWZ * 06RWZ) + (~06NWZ * ~06RWZ * 06RWC * (~SYNC + ~VCOR + SL.IN.06RWCR))	TO	06RWZ;

ASSIGN	06NWC * ~06RWZ * 06NWZ	TO	SL.OUT.06NWCR;
ASSIGN	(06NWCR.S_R * ~SYNC) + (((SL.OUT.06NWCR * SL.IN.06NWCR) + (~SL.OUT.06NWCR * ~SL.IN.06NWCR)) * VCOR * SYNC.WAIT)	TO	06NWCR.S_R;
ASSIGN	SL.OUT.06NWCR * SYNC	TO	06NWCR;
ASSIGN	06RWC * ~06NWZ * 06RWZ	TO	SL.OUT.06RWCR;
ASSIGN	(06RWCR.S_R * ~SYNC) + (((SL.OUT.06RWCR * SL.IN.06RWCR) + (~SL.OUT.06RWCR * ~SL.IN.06RWCR)) * VCOR * SYNC.WAIT)	TO	06RWCR.S_R;
ASSIGN	SL.OUT.06RWCR * SYNC	TO	06RWCR;
ASSIGN	~SL.OUT.07NWCR * ~SL.OUT.07RWCR * 07NWZ * SL.OUT.07LS	TO	SL.OUT.07NL;
ASSIGN	(07NL.S_R * ~SYNC) + (((SL.OUT.07NL * SL.IN.07NL) + (~SL.OUT.07NL * ~SL.IN.07NL)) * VCOR * SYNC.WAIT)	TO	07NL.S_R;
ASSIGN	SL.OUT.07NL * SYNC * (SL.IN.07NL + ~SL.IN.SYNC + ~VCOR)	TO	07NL;
ASSIGN	~SL.OUT.07NWCR * ~SL.OUT.07RWCR * 07RWZ * SL.OUT.07LS	TO	SL.OUT.07RL;
ASSIGN	(07RL.S_R * ~SYNC) + (((SL.OUT.07RL * SL.IN.07RL) + (~SL.OUT.07RL * ~SL.IN.07RL)) * VCOR * SYNC.WAIT)	TO	07RL.S_R;
ASSIGN	SL.OUT.07RL * SYNC * (SL.IN.07RL + ~SL.IN.SYNC + ~VCOR)	TO	07RL;
ASSIGN	(~07RLP * 07NLP * SL.OUT.07LS) + (~07RWZ * 07NWZ) + (~07NWZ * ~07RWZ * 07NWC * (~SYNC + ~VCOR + SL.IN.07NWCR))	TO	07NWZ;
ASSIGN	(~07NLP * 07RLP * SL.OUT.07LS) + (~07NWZ * 07RWZ) + (~07NWZ * ~07RWZ * 07RWC * (~SYNC + ~VCOR + SL.IN.07RWCR))	TO	07RWZ;
ASSIGN	07NWC * ~07RWZ * 07NWZ	TO	SL.OUT.07NWCR;
ASSIGN	(07NWCR.S_R * ~SYNC) + (((SL.OUT.07NWCR * SL.IN.07NWCR) + (~SL.OUT.07NWCR * ~SL.IN.07NWCR)) * VCOR * SYNC.WAIT)	TO	07NWCR.S_R;
ASSIGN	SL.OUT.07NWCR * SYNC	TO	07NWCR;

ASSIGN	07RWC * ~07NWZ * 07RWZ	TO	SL.OUT.07RWCR;
ASSIGN	(07RWCR.S_R * ~SYNC) + (((SL.OUT.07RWCR * SL.IN.07RWCR) + (~SL.OUT.07RWCR * ~SL.IN.07RWCR)) * VCOR * SYNC.WAIT)	TO	07RWCR.S_R;
ASSIGN	SL.OUT.07RWCR * SYNC	TO	07RWCR;
ASSIGN	~SL.OUT.08NWCR * ~SL.OUT.08RWCR * 08NWZ * SL.OUT.08LS	TO	SL.OUT.08NL;
ASSIGN	(08NL.S_R * ~SYNC) + (((SL.OUT.08NL * SL.IN.08NL) + (~SL.OUT.08NL * ~SL.IN.08NL)) * VCOR * SYNC.WAIT)	TO	08NL.S_R;
ASSIGN	SL.OUT.08NL * SYNC * (SL.IN.08NL + ~SL.IN.SYNC + ~VCOR)	TO	08NL;
ASSIGN	~SL.OUT.08NWCR * ~SL.OUT.08RWCR * 08RWZ * SL.OUT.08LS	TO	SL.OUT.08RL;
ASSIGN	(08RL.S_R * ~SYNC) + (((SL.OUT.08RL * SL.IN.08RL) + (~SL.OUT.08RL * ~SL.IN.08RL)) * VCOR * SYNC.WAIT)	TO	08RL.S_R;
ASSIGN	SL.OUT.08RL * SYNC * (SL.IN.08RL + ~SL.IN.SYNC + ~VCOR)	TO	08RL;
ASSIGN	(~08RLP * 08NLP * SL.OUT.08LS) + (~08RWZ * 08NWZ) + (~08NWZ * ~08RWZ * 08NWC * (~SYNC + ~VCOR + SL.IN.08NWCR))	TO	08NWZ;
ASSIGN	(~08NLP * 08RLP * SL.OUT.08LS) + (~08NWZ * 08RWZ) + (~08NWZ * ~08RWZ * 08RWC * (~SYNC + ~VCOR + SL.IN.08RWCR))	TO	08RWZ;
ASSIGN	08NWC * ~08RWZ * 08NWZ	TO	SL.OUT.08NWCR;
ASSIGN	(08NWCR.S_R * ~SYNC) + (((SL.OUT.08NWCR * SL.IN.08NWCR) + (~SL.OUT.08NWCR * ~SL.IN.08NWCR)) * VCOR * SYNC.WAIT)	TO	08NWCR.S_R;
ASSIGN	SL.OUT.08NWCR * SYNC	TO	08NWCR;
ASSIGN	08RWC * ~08NWZ * 08RWZ	TO	SL.OUT.08RWCR;
ASSIGN	(08RWCR.S_R * ~SYNC) + (((SL.OUT.08RWCR * SL.IN.08RWCR) + (~SL.OUT.08RWCR * ~SL.IN.08RWCR)) * VCOR * SYNC.WAIT)	TO	08RWCR.S_R;
ASSIGN	SL.OUT.08RWCR * SYNC	TO	08RWCR;

ASSIGN	~SL.OUT.09NWCR * ~SL.OUT.09RWCR * 09NWZ * SL.OUT.09LS	TO	SL.OUT.09NL;
ASSIGN	(09NL.S_R * ~SYNC) + (((SL.OUT.09NL * SL.IN.09NL) + (~SL.OUT.09NL * ~SL.IN.09NL)) * VCOR * SYNC.WAIT)	TO	09NL.S_R;
ASSIGN	SL.OUT.09NL * SYNC * (SL.IN.09NL + ~SL.IN.SYNC + ~VCOR)	TO	09NL;
ASSIGN	~SL.OUT.09NWCR * ~SL.OUT.09RWCR * 09RWZ * SL.OUT.09LS	TO	SL.OUT.09RL;
ASSIGN	(09RL.S_R * ~SYNC) + (((SL.OUT.09RL * SL.IN.09RL) + (~SL.OUT.09RL * ~SL.IN.09RL)) * VCOR * SYNC.WAIT)	TO	09RL.S_R;
ASSIGN	SL.OUT.09RL * SYNC * (SL.IN.09RL + ~SL.IN.SYNC + ~VCOR)	TO	09RL;
ASSIGN	(~09RLP * 09NLP * SL.OUT.09LS) + (~09RWZ * 09NWZ) + (~09NWZ * ~09RWZ * 09NWC * (~SYNC + ~VCOR + SL.IN.09NWCR))	TO	09NWZ;
ASSIGN	(~09NLP * 09RLP * SL.OUT.09LS) + (~09NWZ * 09RWZ) + (~09NWZ * ~09RWZ * 09RWC * (~SYNC + ~VCOR + SL.IN.09RWCR))	TO	09RWZ;
ASSIGN	09NWC * ~09RWZ * 09NWZ	TO	SL.OUT.09NWCR;
ASSIGN	(09NWCR.S_R * ~SYNC) + (((SL.OUT.09NWCR * SL.IN.09NWCR) + (~SL.OUT.09NWCR * ~SL.IN.09NWCR)) * VCOR * SYNC.WAIT)	TO	09NWCR.S_R;
ASSIGN	SL.OUT.09NWCR * SYNC	TO	09NWCR;
ASSIGN	09RWC * ~09NWZ * 09RWZ	TO	SL.OUT.09RWCR;
ASSIGN	(09RWCR.S_R * ~SYNC) + (((SL.OUT.09RWCR * SL.IN.09RWCR) + (~SL.OUT.09RWCR * ~SL.IN.09RWCR)) * VCOR * SYNC.WAIT)	TO	09RWCR.S_R;
ASSIGN	SL.OUT.09RWCR * SYNC	TO	09RWCR;
ASSIGN	~SL.OUT.10NWCR * ~SL.OUT.10RWCR * 10NWZ * SL.OUT.10LS	TO	SL.OUT.10NL;
ASSIGN	(10NL.S_R * ~SYNC) + (((SL.OUT.10NL * SL.IN.10NL) + (~SL.OUT.10NL * ~SL.IN.10NL)) * VCOR * SYNC.WAIT)	TO	10NL.S_R;
ASSIGN	SL.OUT.10NL * SYNC * (SL.IN.10NL + ~SL.IN.SYNC + ~VCOR)	TO	10NL;

ASSIGN	~SL.OUT.10NWCR * ~SL.OUT.10RWCR * 10RWZ * SL.OUT.10LS	TO	SL.OUT.10RL;
ASSIGN	(10RL.S_R * ~SYNC) + (((SL.OUT.10RL * SL.IN.10RL) + (~SL.OUT.10RL * ~SL.IN.10RL)) * VCOR * SYNC.WAIT)	TO	10RL.S_R;
ASSIGN	SL.OUT.10RL * SYNC * (SL.IN.10RL + ~SL.IN.SYNC + ~VCOR)	TO	10RL;
ASSIGN	(~10RLP * 10NLP * SL.OUT.10LS) + (~10RWZ * 10NWZ) + (~10NWZ * ~10RWZ * 10NWC * (~SYNC + ~VCOR + SL.IN.10NWCR))	TO	10NWZ;
ASSIGN	(~10NLP * 10RLP * SL.OUT.10LS) + (~10NWZ * 10RWZ) + (~10NWZ * ~10RWZ * 10RWC * (~SYNC + ~VCOR + SL.IN.10RWCR))	TO	10RWZ;
ASSIGN	10NWC * ~10RWZ * 10NWZ	TO	SL.OUT.10NWCR;
ASSIGN	(10NWCR.S_R * ~SYNC) + (((SL.OUT.10NWCR * SL.IN.10NWCR) + (~SL.OUT.10NWCR * ~SL.IN.10NWCR)) * VCOR * SYNC.WAIT)	TO	10NWCR.S_R;
ASSIGN	SL.OUT.10NWCR * SYNC	TO	10NWCR;
ASSIGN	10RWC * ~10NWZ * 10RWZ	TO	SL.OUT.10RWCR;
ASSIGN	(10RWCR.S_R * ~SYNC) + (((SL.OUT.10RWCR * SL.IN.10RWCR) + (~SL.OUT.10RWCR * ~SL.IN.10RWCR)) * VCOR * SYNC.WAIT)	TO	10RWCR.S_R;
ASSIGN	SL.OUT.10RWCR * SYNC	TO	10RWCR;
ASSIGN	~SL.OUT.11NWCR * ~SL.OUT.11RWCR * 11NWZ * SL.OUT.11LS	TO	SL:OUT.11NL;
ASSIGN	(11NL.S_R * ~SYNC) + (((SL.OUT.11NL * SL.IN.11NL) + (~SL.OUT.11NL * ~SL.IN.11NL)) * VCOR * SYNC.WAIT)	TO	11NL.S_R;
ASSIGN	SL.OUT.11NL * SYNC * (SL.IN.11NL + ~SL.IN.SYNC + ~VCOR)	TO	11NL;
ASSIGN	~SL.OUT.11NWCR * ~SL.OUT.11RWCR * 11RWZ * SL.OUT.11LS	TO	SL.OUT.11RL;
ASSIGN	(11RL.S_R * ~SYNC) + (((SL.OUT.11RL * SL.IN.11RL) + (~SL.OUT.11RL * ~SL.IN.11RL)) * VCOR * SYNC.WAIT)	TO	11RL.S_R;
ASSIGN	SL.OUT.11RL * SYNC * (SL.IN.11RL + ~SL.IN.SYNC + ~VCOR)	TO	11RL;

ASSIGN	(~11RLP * 11NLP * SL.OUT.11LS) + (~11RWZ * 11NWZ) + (~11NWZ * ~11RWZ * 11NWC * (~SYNC + ~VCOR + SL.IN.11NWCR))	TO	11NWZ;
ASSIGN	(~11NLP * 11RLP * SL.OUT.11LS) + (~11NWZ * 11RWZ) + (~11NWZ * ~11RWZ * 11RWC * (~SYNC + ~VCOR + SL.IN.11RWCR))	TO	11RWZ;
ASSIGN	11NWC * ~11RWZ * 11NWZ	TO	SL.OUT.11NWCR;
ASSIGN	(11NWCR.S_R * ~SYNC) + (((SL.OUT.11NWCR * SL.IN.11NWCR) + (~SL.OUT.11NWCR * ~SL.IN.11NWCR)) * VCOR * SYNC.WAIT)	TO	11NWCR.S_R;
ASSIGN	SL.OUT.11NWCR * SYNC	TO	11NWCR;
ASSIGN	11RWC * ~11NWZ * 11RWZ	TO	SL.OUT.11RWCR;
ASSIGN	(11RWCR.S_R * ~SYNC) + (((SL.OUT.11RWCR * SL.IN.11RWCR) + (~SL.OUT.11RWCR * ~SL.IN.11RWCR)) * VCOR * SYNC.WAIT)	TO	11RWCR.S_R;
ASSIGN	SL.OUT.11RWCR * SYNC	TO	11RWCR;
ASSIGN	~SL.OUT.12NWCR * ~SL.OUT.12RWCR * 12NWZ * SL.OUT.12LS	TO	SL.OUT.12NL;
ASSIGN	(12NL.S_R * ~SYNC) + (((SL.OUT.12NL * SL.IN.12NL) + (~SL.OUT.12NL * ~SL.IN.12NL)) * VCOR * SYNC.WAIT)	TO	12NL.S_R;
ASSIGN	SL.OUT.12NL * SYNC * (SL.IN.12NL + ~SL.IN.SYNC + ~VCOR)	TO	12NL;
ASSIGN	~SL.OUT.12NWCR * ~SL.OUT.12RWCR * 12RWZ * SL.OUT.12LS	TO	SL.OUT.12RL;
ASSIGN	(12RL.S_R * ~SYNC) + (((SL.OUT.12RL * SL.IN.12RL) + (~SL.OUT.12RL * ~SL.IN.12RL)) * VCOR * SYNC.WAIT)	TO	12RL.S_R;
ASSIGN	SL.OUT.12RL * SYNC * (SL.IN.12RL + ~SL.IN.SYNC + ~VCOR)	TO	12RL;
ASSIGN	(~12RLP * 12NLP * SL.OUT.12LS) + (~12RWZ * 12NWZ) + (~12NWZ * ~12RWZ * 12NWC * (~SYNC + ~VCOR + SL.IN.12NWCR))	TO	12NWZ;
ASSIGN	(~12NLP * 12RLP * SL.OUT.12LS) + (~12NWZ * 12RWZ) + (~12NWZ * ~12RWZ * 12RWC * (~SYNC + ~VCOR + SL.IN.12RWCR))	TO	12RWZ;
ASSIGN	12NWC * ~12RWZ * 12NWZ	TO	SL.OUT.12NWCR;

ASSIGN	(12NWCR.S_R * ~SYNC) + (((SL.OUT.12NWCR * SL.IN.12NWCR) + (~SL.OUT.12NWCR * ~SL.IN.12NWCR)) * VCOR * SYNC.WAIT)	TO	12NWCR.S_R;
ASSIGN	SL.OUT.12NWCR * SYNC	TO	12NWCR;
ASSIGN	12RWC * ~12NWZ * 12RWZ	TO	SL.OUT.12RWCR;
ASSIGN	(12RWCR.S_R * ~SYNC) + (((SL.OUT.12RWCR * SL.IN.12RWCR) + (~SL.OUT.12RWCR * ~SL.IN.12RWCR)) * VCOR * SYNC.WAIT)	TO	12RWCR.S_R;
ASSIGN	SL.OUT.12RWCR * SYNC	TO	12RWCR, 12RWCRZ;
ASSIGN	~SL.OUT.13NWCR * ~SL.OUT.13RWCR * 13NWZ * SL.OUT.13LS	TO	SL.OUT.13NL;
ASSIGN	(13NL.S_R * ~SYNC) + (((SL.OUT.13NL * SL.IN.13NL) + (~SL.OUT.13NL * ~SL.IN.13NL)) * VCOR * SYNC.WAIT)	TO	13NL.S_R;
ASSIGN	SL.OUT.13NL * SYNC * (SL.IN.13NL + ~SL.IN.SYNC + ~VCOR)	TO	13NL;
ASSIGN	~SL.OUT.13NWCR * ~SL.OUT.13RWCR * 13RWZ * SL.OUT.13LS	TO	SL.OUT.13RL;
ASSIGN	(13RL.S_R * ~SYNC) + (((SL.OUT.13RL * SL.IN.13RL) + (~SL.OUT.13RL * ~SL.IN.13RL)) * VCOR * SYNC.WAIT)	TO	13RL.S_R;
ASSIGN	SL.OUT.13RL * SYNC * (SL.IN.13RL + ~SL.IN.SYNC + ~VCOR)	TO	13RL;
ASSIGN	(~13RLP * 13NLP * SL.OUT.13LS) + (~13RWZ * 13NWZ) + (~13NWZ * ~13RWZ * 13NWC * (~SYNC + ~VCOR + SL.IN.13NWCR))	TO	13NWZ;
ASSIGN	(~13NLP * 13RLP * SL.OUT.13LS) + (~13NWZ * 13RWZ) + (~13NWZ * ~13RWZ * 13RWC * (~SYNC + ~VCOR + SL.IN.13RWCR))	TO	13RWZ;
ASSIGN	13NWC * ~13RWZ * 13NWZ	TO	SL.OUT.13NWCR;
ASSIGN	(13NWCR.S_R * ~SYNC) + (((SL.OUT.13NWCR * SL.IN.13NWCR) + (~SL.OUT.13NWCR * ~SL.IN.13NWCR)) * VCOR * SYNC.WAIT)	TO	13NWCR.S_R;
ASSIGN	SL.OUT.13NWCR * SYNC	TO	13NWCR;
ASSIGN	13RWC * ~13NWZ * 13RWZ	TO	SL.OUT.13RWCR;

ASSIGN	(13RWCR.S_R * ~SYNC) + (((SL.OUT.13RWCR * SL.IN.13RWCR) + (~SL.OUT.13RWCR * ~SL.IN.13RWCR)) * VCOR * SYNC.WAIT)	TO	13RWCR.S_R;
ASSIGN	SL.OUT.13RWCR * SYNC	TO	13RWCR;
ASSIGN	~SL.OUT.14NWCR * ~SL.OUT.14RWCR * 14NWZ * SL.OUT.14LS	TO	SL.OUT.14NL;
ASSIGN	(14NL.S_R * ~SYNC) + (((SL.OUT.14NL * SL.IN.14NL) + (~SL.OUT.14NL * ~SL.IN.14NL)) * VCOR * SYNC.WAIT)	TO	14NL.S_R;
ASSIGN	SL.OUT.14NL * SYNC * (SL.IN.14NL + ~SL.IN.SYNC + ~VCOR)	TO	14NL;
ASSIGN	~SL.OUT.14NWCR * ~SL.OUT.14RWCR * 14RWZ * SL.OUT.14LS	TO	SL.OUT.14RL;
ASSIGN	(14RL.S_R * ~SYNC) + (((SL.OUT.14RL * SL.IN.14RL) + (~SL.OUT.14RL * ~SL.IN.14RL)) * VCOR * SYNC.WAIT)	TO	14RL.S_R;
ASSIGN	SL.OUT.14RL * SYNC * (SL.IN.14RL + ~SL.IN.SYNC + ~VCOR)	TO	14RL;
ASSIGN	(~14RLP * 14NLP * SL.OUT.14LS) + (~14RWZ * 14NWZ) + (~14NWZ * ~14RWZ * 14NWC * (~SYNC + ~VCOR + SL.IN.14NWCR))	TO	14NWZ;
ASSIGN	(~14NLP * 14RLP * SL.OUT.14LS) + (~14NWZ * 14RWZ) + (~14NWZ * ~14RWZ * 14RWC * (~SYNC + ~VCOR + SL.IN.14RWCR))	TO	14RWZ;
ASSIGN	14NWC * ~14RWZ * 14NWZ	TO	SL.OUT.14NWCR;
ASSIGN	(14NWCR.S_R * ~SYNC) + (((SL.OUT.14NWCR * SL.IN.14NWCR) + (~SL.OUT.14NWCR * ~SL.IN.14NWCR)) * VCOR * SYNC.WAIT)	TO	14NWCR.S_R;
ASSIGN	SL.OUT.14NWCR * SYNC	TO	14NWCR;
ASSIGN	14RWC * ~14NWZ * 14RWZ	TO	SL.OUT.14RWCR;
ASSIGN	(14RWCR.S_R * ~SYNC) + (((SL.OUT.14RWCR * SL.IN.14RWCR) + (~SL.OUT.14RWCR * ~SL.IN.14RWCR)) * VCOR * SYNC.WAIT)	TO	14RWCR.S_R;
ASSIGN	SL.OUT.14RWCR * SYNC	TO	14RWCR;
ASSIGN	33NWCRZ	TO	SL.OUT.33NWCR;

ASSIGN	(33NWCR.S_R * ~SYNC) + (((SL.OUT.33NWCR * SL.IN.33NWCR) + (~SL.OUT.33NWCR * ~SL.IN.33NWCR)) * VCOR * SYNC.WAIT)	TO	33NWCR.S_R;
ASSIGN	SL.OUT.33NWCR * SYNC	TO	33NWCR;

/* **LOCK STICKS** */

ASSIGN	04TPS * 05TPS * 12TPS * 04WS * 04ES * 05ES * 05WS * 12WS * ((01NWZ + ~01NLP) * (01RWZ + ~01RLP) + SL.OUT.01LS)	TO	SL.OUT.01LS;
ASSIGN	(01LS.S_R * ~SYNC) + (((SL.OUT.01LS * SL.IN.01LS) + (~SL.OUT.01LS * ~SL.IN.01LS)) * VCOR * SYNC.WAIT)	TO	01LS.S_R;
ASSIGN	SL.OUT.01LS * SYNC * (SL.IN.01LS + ~SL.IN.SYNC + ~VCOR)	TO	01LS;
ASSIGN	01TPS * 03TPS * 01WS * 01ES * 03WS * 03ES * ((02NWZ + ~02NLP) * (02RWZ + ~02RLP) + SL.OUT.02LS)	TO	SL.OUT.02LS;
ASSIGN	(02LS.S_R * ~SYNC) + (((SL.OUT.02LS * SL.IN.02LS) + (~SL.OUT.02LS * ~SL.IN.02LS)) * VCOR * SYNC.WAIT)	TO	02LS.S_R;
ASSIGN	SL.OUT.02LS * SYNC * (SL.IN.02LS + ~SL.IN.SYNC + ~VCOR)	TO	02LS;
ASSIGN	03TPS * 04TPS * 16TPS * 03WS * 03ES * 04ES * 04WS * 16WS * 16ES * ((03NWZ + ~03NLP) * (03RWZ + ~03RLP) + SL.OUT.03LS)	TO	SL.OUT.03LS;
ASSIGN	(03LS.S_R * ~SYNC) + (((SL.OUT.03LS * SL.IN.03LS) + (~SL.OUT.03LS * ~SL.IN.03LS)) * VCOR * SYNC.WAIT)	TO	03LS.S_R;
ASSIGN	SL.OUT.03LS * SYNC * (SL.IN.03LS + ~SL.IN.SYNC + ~VCOR)	TO	03LS;
ASSIGN	12TPS * 12WS * 12ES * ((04NWZ + ~04NLP) * (04RWZ + ~04RLP) + SL.OUT.04LS)	TO	SL.OUT.04LS;
ASSIGN	(04LS.S_R * ~SYNC) + (((SL.OUT.04LS * SL.IN.04LS) + (~SL.OUT.04LS * ~SL.IN.04LS)) * VCOR * SYNC.WAIT)	TO	04LS.S_R;
ASSIGN	SL.OUT.04LS * SYNC * (SL.IN.04LS + ~SL.IN.SYNC + ~VCOR)	TO	04LS;
ASSIGN	05TPS * 06TPS * 07TPS * 05WS * 05ES * 06ES * 06WS * 07WS * ((05NWZ + ~05NLP) * (05RWZ + ~05RLP) + SL.OUT.05LS)	TO	SL.OUT.05LS;

ASSIGN	(05LS.S_R * ~SYNC) + (((SL.OUT.05LS * SL.IN.05LS) + (~SL.OUT.05LS * ~SL.IN.05LS)) * VCOR * SYNC.WAIT)	TO	05LS.S_R;
ASSIGN	SL.OUT.05LS * SYNC * (SL.IN.05LS + ~SL.IN.SYNC + ~VCOR)	TO	05LS;
ASSIGN	13TPS * 14TPS * 13WS * 13ES * 14ES * 14WS * ((06NWZ + ~06NLP) * (06RWZ + ~06RLP) + SL.OUT.06LS)	TO	SL.OUT.06LS;
ASSIGN	(06LS.S_R * ~SYNC) + (((SL.OUT.06LS * SL.IN.06LS) + (~SL.OUT.06LS * ~SL.IN.06LS)) * VCOR * SYNC.WAIT)	TO	06LS.S_R;
ASSIGN	SL.OUT.06LS * SYNC * (SL.IN.06LS + ~SL.IN.SYNC + ~VCOR)	TO	06LS;
ASSIGN	07TPS * 08TPS * 11TPS * 07WS * 07ES * 08ES * 08WS * 11ES * 11WS * ((07NWZ + ~07NLP) * (07RWZ + ~07RLP) + SL.OUT.07LS)	TO	SL.OUT.07LS;
ASSIGN	(07LS.S_R * ~SYNC) + (((SL.OUT.07LS * SL.IN.07LS) + (~SL.OUT.07LS * ~SL.IN.07LS)) * VCOR * SYNC.WAIT)	TO	07LS.S_R;
ASSIGN	SL.OUT.07LS * SYNC * (SL.IN.07LS + ~SL.IN.SYNC + ~VCOR)	TO	07LS;
ASSIGN	06TPS * 06WS * 06ES * ((08NWZ + ~08NLP) * (08RWZ + ~08RLP) + SL.OUT.08LS)	TO	SL.OUT.08LS;
ASSIGN	(08LS.S_R * ~SYNC) + (((SL.OUT.08LS * SL.IN.08LS) + (~SL.OUT.08LS * ~SL.IN.08LS)) * VCOR * SYNC.WAIT)	TO	08LS.S_R;
ASSIGN	SL.OUT.08LS * SYNC * (SL.IN.08LS + ~SL.IN.SYNC + ~VCOR)	TO	08LS;
ASSIGN	14TPS * 14WS * 14ES * ((09NWZ + ~09NLP) * (09RWZ + ~09RLP) + SL.OUT.09LS)	TO	SL.OUT.09LS;
ASSIGN	(09LS.S_R * ~SYNC) + (((SL.OUT.09LS * SL.IN.09LS) + (~SL.OUT.09LS * ~SL.IN.09LS)) * VCOR * SYNC.WAIT)	TO	09LS.S_R;
ASSIGN	SL.OUT.09LS * SYNC * (SL.IN.09LS + ~SL.IN.SYNC + ~VCOR)	TO	09LS;
ASSIGN	11TPS * 11WS * 11ES * ((10NWZ + ~10NLP) * (10RWZ + ~10RLP) + SL.OUT.10LS)	TO	SL.OUT.10LS;
ASSIGN	(10LS.S_R * ~SYNC) + (((SL.OUT.10LS * SL.IN.10LS) + (~SL.OUT.10LS * ~SL.IN.10LS)) * VCOR * SYNC.WAIT)	TO	10LS.S_R;
ASSIGN	SL.OUT.10LS * SYNC * (SL.IN.10LS + ~SL.IN.SYNC + ~VCOR)	TO	10LS;

ASSIGN	08TPS * 09TPS * 10TPS * 08WS * 08ES * 09ES * 09WS * 10ES * 10WS * ((11NWZ + ~11NLP) * (11RWZ + ~11RLP) + SL.OUT.11LS)	TO	SL.OUT.11LS;
ASSIGN	(11LS.S_R * ~SYNC) + (((SL.OUT.11LS * SL.IN.11LS) + (~SL.OUT.11LS * ~SL.IN.11LS)) * VCOR * SYNC.WAIT)	TO	11LS.S_R;
ASSIGN	SL.OUT.11LS * SYNC * (SL.IN.11LS + ~SL.IN.SYNC + ~VCOR)	TO	11LS;
ASSIGN	10TPS * 10WS * 10ES * ((12NWZ + ~12NLP) * (12RWZ + ~12RLP) + SL.OUT.12LS)	TO	SL.OUT.12LS;
ASSIGN	(12LS.S_R * ~SYNC) + (((SL.OUT.12LS * SL.IN.12LS) + (~SL.OUT.12LS * ~SL.IN.12LS)) * VCOR * SYNC.WAIT)	TO	12LS.S_R;
ASSIGN	SL.OUT.12LS * SYNC * (SL.IN.12LS + ~SL.IN.SYNC + ~VCOR)	TO	12LS;
ASSIGN	09TPS * 09WS * 09ES * ((13NWZ + ~13NLP) * (13RWZ + ~13RLP) + SL.OUT.13LS)	TO	SL.OUT.13LS;
ASSIGN	(13LS.S_R * ~SYNC) + (((SL.OUT.13LS * SL.IN.13LS) + (~SL.OUT.13LS * ~SL.IN.13LS)) * VCOR * SYNC.WAIT)	TO	13LS.S_R;
ASSIGN	SL.OUT.13LS * SYNC * (SL.IN.13LS + ~SL.IN.SYNC + ~VCOR)	TO	13LS;
ASSIGN	09TPS * 09WS * 09ES * ((14NWZ + ~14NLP) * (14RWZ + ~14RLP) + SL.OUT.14LS)	TO	SL.OUT.14LS;
ASSIGN	(14LS.S_R * ~SYNC) + (((SL.OUT.14LS * SL.IN.14LS) + (~SL.OUT.14LS * ~SL.IN.14LS)) * VCOR * SYNC.WAIT)	TO	14LS.S_R;
ASSIGN	SL.OUT.14LS * SYNC * (SL.IN.14LS + ~SL.IN.SYNC + ~VCOR)	TO	14LS;
ASSIGN	(01NWZ * ~SL.OUT.01NWCR * SL.OUT.01NL) + (01RWZ * ~SL.OUT.01RWCR * SL.OUT.01RL)	TO	01LSR;
ASSIGN	(02NWZ * ~SL.OUT.02NWCR * SL.OUT.02NL) + (02RWZ * ~SL.OUT.02RWCR * SL.OUT.02RL)	TO	02LSR;
ASSIGN	(03NWZ * ~SL.OUT.03NWCR * SL.OUT.03NL) + (03RWZ * ~SL.OUT.03RWCR * SL.OUT.03RL)	TO	03LSR;
ASSIGN	(04NWZ * ~SL.OUT.04NWCR * SL.OUT.04NL) + (04RWZ * ~SL.OUT.04RWCR * SL.OUT.04RL)	TO	04LSR;
ASSIGN	(05NWZ * ~SL.OUT.05NWCR * SL.OUT.05NL) + (05RWZ * ~SL.OUT.05RWCR * SL.OUT.05RL)	TO	05LSR;

ASSIGN	(06NWZ * ~SL.OUT.06NWCR * SL.OUT.06NL) + (06RWZ * ~SL.OUT.06RWCR * SL.OUT.06RL)	TO	06LSR;
ASSIGN	(07NWZ * ~SL.OUT.07NWCR * SL.OUT.07NL) + (07RWZ * ~SL.OUT.07RWCR * SL.OUT.07RL)	TO	07LSR;
ASSIGN	(08NWZ * ~SL.OUT.08NWCR * SL.OUT.08NL) + (08RWZ * ~SL.OUT.08RWCR * SL.OUT.08RL)	TO	08LSR;
ASSIGN	(09NWZ * ~SL.OUT.09NWCR * SL.OUT.09NL) + (09RWZ * ~SL.OUT.09RWCR * SL.OUT.09RL)	TO	09LSR;
ASSIGN	(10NWZ * ~SL.OUT.10NWCR * SL.OUT.10NL) + (10RWZ * ~SL.OUT.10RWCR * SL.OUT.10RL)	TO	10LSR;
ASSIGN	(11NWZ * ~SL.OUT.11NWCR * SL.OUT.11NL) + (11RWZ * ~SL.OUT.11RWCR * SL.OUT.11RL)	TO	11LSR;
ASSIGN	(12NWZ * ~SL.OUT.12NWCR * SL.OUT.12NL) + (12RWZ * ~SL.OUT.12RWCR * SL.OUT.12RL)	TO	12LSR;
ASSIGN	(13NWZ * ~SL.OUT.13NWCR * SL.OUT.13NL) + (13RWZ * ~SL.OUT.13RWCR * SL.OUT.13RL)	TO	13LSR;
ASSIGN	(14NWZ * ~SL.OUT.14NWCR * SL.OUT.14NL) + (14RWZ * ~SL.OUT.14RWCR * SL.OUT.14RL)	TO	14LSR;

/* **APPROACH STICKS AND TIMERS** */

ASSIGN	~SL.OUT.501H * ~501R * (501_540TE + ~01TPS * POPS + SL.OUT.501AS)	TO	SL.OUT.501AS;
ASSIGN	(501AS.S_R * ~SYNC) + (((SL.OUT.501AS * SL.IN.501AS) + (~SL.OUT.501AS * ~SL.IN.501AS)) * VCOR * SYNC.WAIT)	TO	501AS.S_R;
ASSIGN	SL.OUT.501AS * SYNC * (SL.IN.501AS + ~SL.IN.SYNC + ~VCOR)	TO	501AS;
ASSIGN	~SL.OUT.540H * ~540R * (501_540TE + ~01TPS * POPS + SL.OUT.540AS)	TO	SL.OUT.540AS;
ASSIGN	(540AS.S_R * ~SYNC) + (((SL.OUT.540AS * SL.IN.540AS) + (~SL.OUT.540AS * ~SL.IN.540AS)) * VCOR * SYNC.WAIT)	TO	540AS.S_R;
ASSIGN	SL.OUT.540AS * SYNC * (SL.IN.540AS + ~SL.IN.SYNC + ~VCOR)	TO	540AS;
ASSIGN	~SL.OUT.501AS * ~SL.OUT.501H * ~501R + ~SL.OUT.540AS * ~SL.OUT.540H * ~540R	TO	501_540TE;

ASSIGN	~SL.OUT.460H * ~SL.OUT.460L * ~460R * (460TE + ~03TPS * POPS + SL.OUT.460AS)	TO	SL.OUT.460AS;
ASSIGN	(460AS.S_R * ~SYNC) + (((SL.OUT.460AS * SL.IN.460AS) + (~SL.OUT.460AS * ~SL.IN.460AS)) * VCOR * SYNC.WAIT)	TO	460AS.S_R;
ASSIGN	SL.OUT.460AS * SYNC * (SL.IN.460AS + ~SL.IN.SYNC + ~VCOR)	TO	460AS;
ASSIGN	~SL.OUT.460AS * ~SL.OUT.460H * ~SL.OUT.460L * ~460R	TO	460TE;
ASSIGN	~SL.OUT.512H * ~512R * (512TE + ~02TPS * POPS + SL.OUT.512AS)	TO	SL.OUT.512AS;
ASSIGN	(512AS.S_R * ~SYNC) + (((SL.OUT.512AS * SL.IN.512AS) + (~SL.OUT.512AS * ~SL.IN.512AS)) * VCOR * SYNC.WAIT)	TO	512AS.S_R;
ASSIGN	SL.OUT.512AS * SYNC * (SL.IN.512AS + ~SL.IN.SYNC + ~VCOR)	TO	512AS;
ASSIGN	~SL.OUT.512AS * ~SL.OUT.512H * ~512R	TO	512TE;
ASSIGN	~SL.OUT.340H * ~340R * (312_340TE + ~16TPS * POPS + SL.OUT.340AS)	TO	SL.OUT.340AS;
ASSIGN	(340AS.S_R * ~SYNC) + (((SL.OUT.340AS * SL.IN.340AS) + (~SL.OUT.340AS * ~SL.IN.340AS)) * VCOR * SYNC.WAIT)	TO	340AS.S_R;
ASSIGN	SL.OUT.340AS * SYNC * (SL.IN.340AS + ~SL.IN.SYNC + ~VCOR)	TO	340AS;
ASSIGN	~SL.OUT.312H * ~312R * (312_340TE + ~16TPS * POPS + SL.OUT.312AS)	TO	SL.OUT.312AS;
ASSIGN	(312AS.S_R * ~SYNC) + (((SL.OUT.312AS * SL.IN.312AS) + (~SL.OUT.312AS * ~SL.IN.312AS)) * VCOR * SYNC.WAIT)	TO	312AS.S_R;
ASSIGN	SL.OUT.312AS * SYNC * (SL.IN.312AS + ~SL.IN.SYNC + ~VCOR)	TO	312AS;
ASSIGN	~SL.OUT.312AS * ~SL.OUT.312H * ~312R + ~SL.OUT.340AS * ~SL.OUT.340H * ~340R	TO	312_340TE;
ASSIGN	~SL.OUT.611H * ~611R * ~SL.OUT.612H * ~612R * (611_612TE + ~06TPS * POPS + SL.OUT.611_612AS)	TO	SL.OUT.611_612AS;
ASSIGN	(611_612AS.S_R * ~SYNC) + (((SL.OUT.611_612AS * SL.IN.611_612AS) + (~SL.OUT.611_612AS * ~SL.IN.611_612AS)) * VCOR * SYNC.WAIT)	TO	611_612AS.S_R;
ASSIGN	SL.OUT.611_612AS * SYNC * (SL.IN.611_612AS + ~SL.IN.SYNC + ~VCOR)	TO	611_612AS;

ASSIGN	~SL.OUT.611_612AS * ~SL.OUT.611H * ~611R * ~SL.OUT.612H * ~612R	TO	611_612TE;
ASSIGN	~SL.OUT.531H * ~531R * ~SL.OUT.532H * ~532R * ~SL.OUT.533H * ~533R * (432_531_534TE + ~09TPS * POPS + SL.OUT.531_532_533AS)	TO	SL.OUT.531_532_533AS;
ASSIGN	(531_532_533AS.S_R * ~SYNC) + (((SL.OUT.531_532_533AS * SL.IN.531_532_533AS) + (~SL.OUT.531_532_533AS * ~SL.IN.531_532_533AS)) * VCOR * SYNC.WAIT)	TO	531_532_533AS.S_R;
ASSIGN	SL.OUT.531_532_533AS * SYNC * (SL.IN.531_532_533AS + ~SL.IN.SYNC + ~VCOR)	TO	531_532_533AS;
ASSIGN	~SL.OUT.432H * ~432R * ~SL.OUT.534H * ~534R * (432_531_534TE + ~10TPS * POPS + SL.OUT.432_534AS)	TO	SL.OUT.432_534AS;
ASSIGN	(432_534AS.S_R * ~SYNC) + (((SL.OUT.432_534AS * SL.IN.432_534AS) + (~SL.OUT.432_534AS * ~SL.IN.432_534AS)) * VCOR * SYNC.WAIT)	TO	432_534AS.S_R;
ASSIGN	SL.OUT.432_534AS * SYNC * (SL.IN.432_534AS + ~SL.IN.SYNC + ~VCOR)	TO	432_534AS;
ASSIGN	~SL.OUT.432_534AS * ~SL.OUT.432H * ~432R * ~SL.OUT.534H * ~534R + ~SL.OUT.531_532_533AS * ~SL.OUT.531H * ~531R * ~SL.OUT.532H * ~532R * ~SL.OUT.533H * ~533R	TO	432_531_534TE;
ASSIGN	~SL.OUT.331H * ~331R * ~SL.OUT.431H * ~431R * (331_431_362TE + ~11TPS * POPS + SL.OUT.331_431AS)	TO	SL.OUT.331_431AS;
ASSIGN	(331_431AS.S_R * ~SYNC) + (((SL.OUT.331_431AS * SL.IN.331_431AS) + (~SL.OUT.331_431AS * ~SL.IN.331_431AS)) * VCOR * SYNC.WAIT)	TO	331_431AS.S_R;
ASSIGN	SL.OUT.331_431AS * SYNC * (SL.IN.331_431AS + ~SL.IN.SYNC + ~VCOR)	TO	331_431AS;
ASSIGN	~SL.OUT.362H * ~SL.OUT.362L * ~362R * (331_431_362TE + ~07TPS * POPS + SL.OUT.362AS)	TO	SL.OUT.362AS;
ASSIGN	(362AS.S_R * ~SYNC) + (((SL.OUT.362AS * SL.IN.362AS) + (~SL.OUT.362AS * ~SL.IN.362AS)) * VCOR * SYNC.WAIT)	TO	362AS.S_R;
ASSIGN	SL.OUT.362AS * SYNC * (SL.IN.362AS + ~SL.IN.SYNC + ~VCOR)	TO	362AS;
ASSIGN	~SL.OUT.331_431AS * ~SL.OUT.331H * ~331R * ~SL.OUT.431H * ~431R + ~SL.OUT.362AS * ~SL.OUT.362H * ~SL.OUT.362L * ~362R	TO	331_431_362TE;
ASSIGN	~SL.OUT.332H * ~332R * ~SL.OUT.333H * ~333R * (332_334TE + ~14TPS * POPS +	TO	SL.OUT.332_333AS;

	SL.OUT.332_333AS)		
ASSIGN	(332_333AS.S_R * ~SYNC) + (((SL.OUT.332_333AS * SL.IN.332_333AS) + (~SL.OUT.332_333AS * ~SL.IN.332_333AS)) * VCOR * SYNC.WAIT)	TO	332_333AS.S_R;
ASSIGN	SL.OUT.332_333AS * SYNC * (SL.IN.332_333AS + ~SL.IN.SYNC + ~VCOR)	TO	332_333AS;
ASSIGN	~SL.OUT.334H * ~334R * (332_334TE + ~13TPS * POPS + SL.OUT.334AS)	TO	SL.OUT.334AS;
ASSIGN	(334AS.S_R * ~SYNC) + (((SL.OUT.334AS * SL.IN.334AS) + (~SL.OUT.334AS * ~SL.IN.334AS)) * VCOR * SYNC.WAIT)	TO	334AS.S_R;
ASSIGN	SL.OUT.334AS * SYNC * (SL.IN.334AS + ~SL.IN.SYNC + ~VCOR)	TO	334AS;
ASSIGN	~SL.OUT.332_333AS * ~SL.OUT.332H * ~332R * ~SL.OUT.333H * ~333R + ~SL.OUT.334AS * ~SL.OUT.334H * ~334R	TO	332_334TE;
ASSIGN	~SL.OUT.311H * ~311R * (311_361TE + ~12TPS * POPS + SL.OUT.311AS)	TO	SL.OUT.311AS;
ASSIGN	(311AS.S_R * ~SYNC) + (((SL.OUT.311AS * SL.IN.311AS) + (~SL.OUT.311AS * ~SL.IN.311AS)) * VCOR * SYNC.WAIT)	TO	311AS.S_R;
ASSIGN	SL.OUT.311AS * SYNC * (SL.IN.311AS + ~SL.IN.SYNC + ~VCOR)	TO	311AS;
ASSIGN	~SL.OUT.361H * ~SL.OUT.361L * ~361R * (311_361TE + ~12TPS * POPS + SL.OUT.361AS)	TO	SL.OUT.361AS;
ASSIGN	(361AS.S_R * ~SYNC) + (((SL.OUT.361AS * SL.IN.361AS) + (~SL.OUT.361AS * ~SL.IN.361AS)) * VCOR * SYNC.WAIT)	TO	361AS.S_R;
ASSIGN	SL.OUT.361AS * SYNC * (SL.IN.361AS + ~SL.IN.SYNC + ~VCOR)	TO	361AS;
ASSIGN	~SL.OUT.311AS * ~SL.OUT.311H * ~311R + ~SL.OUT.361AS * ~SL.OUT.361H * ~SL.OUT.361L * ~361R	TO	311_361TE;

/* DIRECTIONAL ROUTE STICKS */

ASSIGN	SL.OUT.540AS * (03ES + SL.OUT.02NWCR) * (01TPS + 01ES)	TO	01ES;
ASSIGN	01ES * (02TPS + 02ES)	TO	02ES;
ASSIGN	SL.OUT.460AS * (03TPS + 03ES)	TO	03ES;

ASSIGN	(03ES + SL.OUT.03RWCR + SL.OUT.02RWCR) * (04TPS + 04ES)	TO	04ES;
ASSIGN	(04ES + SL.OUT.01NWCR) * (05TPS + 05ES)	TO	05ES;
ASSIGN	(05ES + SL.OUT.05NWCR) * (06TPS + 06ES)	TO	06ES;
ASSIGN	SL.OUT.362AS * (07TPS + 07ES)	TO	07ES;
ASSIGN	(07ES + SL.OUT.07NWCR) * (08TPS + 08ES)	TO	08ES;
ASSIGN	(08ES + SL.OUT.11NWCR) * (09TPS + 09ES)	TO	09ES;
ASSIGN	(08ES + SL.OUT.11RWCR) * (10TPS + 10ES)	TO	10ES;
ASSIGN	(07ES + SL.OUT.07RWCR) * (11TPS + 11ES)	TO	11ES;
ASSIGN	SL.OUT.361AS * (12TPS + 12ES)	TO	12ES;
ASSIGN	(12ES + SL.OUT.04NWCR) * (13TPS + 13ES)	TO	13ES;
ASSIGN	(13ES + SL.OUT.06RWCR) * (14TPS + 14ES)	TO	14ES;
ASSIGN	(12ES + SL.OUT.04RWCR) * (SL.OUT.15TS + SL.OUT.15ES)	TO	SL.OUT.15ES;
ASSIGN	(15ES.S_R * ~SYNC) + (((SL.OUT.15ES * SL.IN.15ES) + (~SL.OUT.15ES * ~SL.IN.15ES)) * VCOR * SYNC.WAIT)	TO	15ES.S_R;
ASSIGN	SL.OUT.15ES * SYNC	TO	15ES, 15ESZ;
ASSIGN	SL.OUT.340AS * (03ES + SL.OUT.03NWCR) * (16TPS + 16ES)	TO	16ES;
ASSIGN	16ES * (SL.OUT.17TS + SL.OUT.17ES)	TO	SL.OUT.17ES;
ASSIGN	(17ES.S_R * ~SYNC) + (((SL.OUT.17ES * SL.IN.17ES) + (~SL.OUT.17ES * ~SL.IN.17ES)) * VCOR * SYNC.WAIT)	TO	17ES.S_R;
ASSIGN	SL.OUT.17ES * SYNC	TO	17ES, 17ESZ;
ASSIGN	(10ES + SL.OUT.12NWCR) * (20TS + SL.OUT.20ES)	TO	SL.OUT.20ES;
ASSIGN	(20ES.S_R * ~SYNC) + (((SL.OUT.20ES * SL.IN.20ES) + (~SL.OUT.20ES * ~SL.IN.20ES)) * VCOR * SYNC.WAIT)	TO	20ES.S_R;

ASSIGN	SL.OUT.20ES * SYNC	TO	20ES, 20ESZ;
ASSIGN	(11ES + SL.OUT.10NWCR) * (21TPS + SL.OUT.21ES)	TO	SL.OUT.21ES;
ASSIGN	(21ES.S_R * ~SYNC) + (((SL.OUT.21ES * SL.IN.21ES) + (~SL.OUT.21ES * ~SL.IN.21ES)) * VCOR * SYNC.WAIT)	TO	21ES.S_R;
ASSIGN	SL.OUT.21ES * SYNC	TO	21ES, 21ESZ;
ASSIGN	SL.OUT.501AS * (01TPS + 01WS)	TO	01WS;
ASSIGN	SL.OUT.512AS * (02TPS + 02WS)	TO	02WS;
ASSIGN	(01WS + SL.OUT.02NWCR) * 04WS * (16WS + SL.OUT.03NWCR) * (03TPS + 03WS)	TO	03WS;
ASSIGN	05WS * 12WS * (04TPS + 04WS)	TO	04WS;
ASSIGN	06WS * 07WS * (05TPS + 05WS)	TO	05WS;
ASSIGN	SL.OUT.611_612AS * (06TPS + 06WS)	TO	06WS;
ASSIGN	11WS * 08WS * (07TPS + 07WS)	TO	07WS;
ASSIGN	09WS * 10WS * (08TPS + 08WS)	TO	08WS;
ASSIGN	SL.OUT.531_532_533AS * (09TPS + 09WS)	TO	09WS;
ASSIGN	SL.OUT.432_534AS * (10TPS + 10WS)	TO	10WS;
ASSIGN	SL.OUT.331_431AS * (11TPS + 11WS)	TO	11WS;
ASSIGN	SL.OUT.311AS * 13WS * (12TPS + 12WS)	TO	12WS;
ASSIGN	SL.OUT.334AS * 14WS * (13TPS + 13WS)	TO	13WS;
ASSIGN	SL.OUT.332_333AS * (14TPS + 14WS)	TO	14WS;
ASSIGN	15WSZ	TO	15WS;
ASSIGN	17WSZ	TO	17WS;
ASSIGN	20WSZ	TO	20WS;

ASSIGN	21WSZ	TO	21WS;
ASSIGN	SL.OUT.312AS * (16TPS + 16WS)	TO	16WS;

/* **TRACK REPEATER STICKS AND TIMERS** */

ASSIGN	01TPS1 * (01TE + 01TPS)	TO	01TPS;
ASSIGN	01TPS1 * ~01TPS	TO	01TE;
ASSIGN	02TPS1 * (02TE + 02TPS)	TO	02TPS;
ASSIGN	02TPS1 * ~02TPS	TO	02TE;
ASSIGN	03TPS1 * (03TE + 03TPS)	TO	03TPS;
ASSIGN	03TPS1 * ~03TPS	TO	03TE;
ASSIGN	04TPS1 * (04TE + 04TPS)	TO	04TPS;
ASSIGN	04TPS1 * ~04TPS	TO	04TE;
ASSIGN	05TPS1 * (05TE + 05TPS)	TO	05TPS;
ASSIGN	05TPS1 * ~05TPS	TO	05TE;
ASSIGN	06TPS1 * (06TE + 06TPS)	TO	06TPS;
ASSIGN	06TPS1 * ~06TPS	TO	06TE;
ASSIGN	07TPS1 * (07TE + 07TPS)	TO	07TPS;
ASSIGN	07TPS1 * ~07TPS	TO	07TE;
ASSIGN	08TPS1 * (08TE + 08TPS)	TO	08TPS;
ASSIGN	08TPS1 * ~08TPS	TO	08TE;
ASSIGN	09TPS1 * (09TE + 09TPS)	TO	09TPS;

ASSIGN	09TPS1 * ~09TPS	TO	09TE;
ASSIGN	10TPS1 * (10TE + 10TPS)	TO	10TPS;
ASSIGN	10TPS1 * ~10TPS	TO	10TE;
ASSIGN	11TPS1 * (11TE + 11TPS)	TO	11TPS;
ASSIGN	11TPS1 * ~11TPS	TO	11TE;
ASSIGN	12TPS1 * (12TE + 12TPS)	TO	12TPS;
ASSIGN	12TPS1 * ~12TPS	TO	12TE;
ASSIGN	13TPS1 * (13TE + 13TPS)	TO	13TPS;
ASSIGN	13TPS1 * ~13TPS	TO	13TE;
ASSIGN	14TPS1 * (14TE + 14TPS)	TO	14TPS;
ASSIGN	14TPS1 * ~14TPS	TO	14TE;
ASSIGN	15TPS1 * (15TE + 15TPS)	TO	15TPS;
ASSIGN	15TPS1 * ~15TPS	TO	15TE;
ASSIGN	16TPS1 * (16TE + 16TPS)	TO	16TPS;
ASSIGN	16TPS1 * ~16TPS	TO	16TE;
ASSIGN	17TPS1 * (17TE + 17TPS)	TO	17TPS;
ASSIGN	17TPS1 * ~17TPS	TO	17TE;
ASSIGN	20TPS1 * (20TE + 20TPS)	TO	20TPS;
ASSIGN	20TPS1 * ~20TPS	TO	20TE;
ASSIGN	21TPS1 * (21TE + 21TPS)	TO	21TPS;
ASSIGN	21TPS1 * ~21TPS	TO	21TE;

ASSIGN	((12TPS * SL.OUT.33NWCR * ~26TPS + ~12TPS * SL.OUT.04NWCR * 26TPS) * ~15TE + SL.OUT.15TS) * 15TPS	TO	SL.OUT.15TS;
ASSIGN	(15TS.S_R * ~SYNC) + (((SL.OUT.15TS * SL.IN.15TS) + (~SL.OUT.15TS * ~SL.IN.15TS)) * VCOR * SYNC.WAIT)	TO	15TS.S_R;
ASSIGN	SL.OUT.15TS * SYNC	TO	15TS, 15TSZ;
ASSIGN	((35TPS * ~16TPS + ~35TPS * 16TPS) * ~17TE + SL.OUT.17TS) * 17TPS	TO	SL.OUT.17TS;
ASSIGN	(17TS.S_R * ~SYNC) + (((SL.OUT.17TS * SL.IN.17TS) + (~SL.OUT.17TS * ~SL.IN.17TS)) * VCOR * SYNC.WAIT)	TO	17TS.S_R;
ASSIGN	SL.OUT.17TS * SYNC	TO	17TS, 17TSZ;
ASSIGN	20TSZ	TO	20TS;
ASSIGN	ONE + 26TPS1 * (26TE + 26TPS)	TO	26TPS;
ASSIGN	26TPS1 * ~26TPS	TO	26TE;
ASSIGN	ONE + 35TPS1 * (35TE + 35TPS)	TO	35TPS;
ASSIGN	35TPS1 * ~35TPS	TO	35TE;

/* SIGNAL CONTROL */

ASSIGN	540R * SL.OUT.02NWCR * ~SL.OUT.02LS * 01TPS * ~01TE * ~01ES * 01WS * 02TPS * ~02TE * ~02ES * 02WS * ~SL.OUT.540AS * ~501_540TE * ~512TE * SL.OUT.512AS * SL.OUT.501AS	TO	SL.OUT.540H;
ASSIGN	(540H.S_R * ~SYNC) + (((SL.OUT.540H * SL.IN.540H) + (~SL.OUT.540H * ~SL.IN.540H)) * VCOR * SYNC.WAIT)	TO	540H.S_R;
ASSIGN	SL.OUT.540H * SYNC * (NORMAL + SL.IN.540H + ~SL.IN.SYNC + ~VCOR)	TO	540H;
ASSIGN	512R * 02TPS * ~02TE * ~02WS * 02ES * 01ES * ~SL.OUT.512AS * ~512TE	TO	SL.OUT.512H;
ASSIGN	(512H.S_R * ~SYNC) + (((SL.OUT.512H * SL.IN.512H) + (~SL.OUT.512H * ~SL.IN.512H))	TO	512H.S_R;

	* VCOR * SYNC.WAIT)		
ASSIGN	SL.OUT.512H * SYNC * (NORMAL + SL.IN.512H + ~SL.IN.SYNC + ~VCOR)	TO	512H;
ASSIGN	460R * 03TPS * ~03TE * 03WS * ~SL.OUT.02LS * ~SL.OUT.460AS * ~460TE	TO	460H1;
ASSIGN	460H1 * (SL.OUT.02RWCR * 01TPS * ~01TE * ~01ES * 02TPS * ~02TE * ~02ES * 02WS * SL.OUT.512AS * ~512TE * SL.OUT.501AS * ~501_540TE + SL.OUT.02NWCR * 04TPS * ~04TE * ~SL.OUT.03LS * (SL.OUT.03NWCR * ~SL.OUT.01LS * (SL.OUT.01RWCR * ~SL.OUT.05LS * SL.OUT.05NWCR * 06TPS * ~06TE * 07WS * ~05ES + SL.OUT.01NWCR * ~04ES * 12WS) * 05TPS * ~05TE * 07TPS * 12TPS + SL.OUT.03RWCR * 16TPS * ~16TE * SL.OUT.17TS * ~SL.OUT.17ES * 17WS * SL.OUT.312AS * ~312_340TE))	TO	SL.OUT.460H;
ASSIGN	(460H.S_R * ~SYNC) + (((SL.OUT.460H * SL.IN.460H) + (~SL.OUT.460H * ~SL.IN.460H)) * VCOR * SYNC.WAIT)	TO	460H.S_R;
ASSIGN	SL.OUT.460H * SYNC * (NORMAL + SL.IN.460H + ~SL.IN.SYNC + ~VCOR)	TO	460H;
ASSIGN	460R * SL.OUT.01RWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * SL.OUT.05RWCR * ~SL.OUT.05LS * ~SL.OUT.08LS * 03TPS * ~03TE * 03WS * 04TPS * ~04TE * 05TPS * ~05TE * 06TPS * ~06TE * ~06ES * 07TPS * 12TPS * ~SL.OUT.460AS * ~460TE * SL.OUT.611_612AS * ~611_612TE * (SL.OUT.08RWCR + SL.OUT.08NWCR)	TO	SL.OUT.460L;
ASSIGN	(460L.S_R * ~SYNC) + (((SL.OUT.460L * SL.IN.460L) + (~SL.OUT.460L * ~SL.IN.460L)) * VCOR * SYNC.WAIT)	TO	460L.S_R;
ASSIGN	SL.OUT.460L * SYNC * (NORMAL + SL.IN.460L + ~SL.IN.SYNC + ~VCOR)	TO	460L;
ASSIGN	501R * ~SL.OUT.02LS * 01TPS * ~01TE * 01ES * ~SL.OUT.501AS * ~501_540TE * (SL.OUT.02NWCR * ~01WS * SL.OUT.540AS + SL.OUT.02RWCR * 03TPS * ~03TE * ~03WS * SL.OUT.460AS * ~460TE)	TO	SL.OUT.501H;
ASSIGN	(501H.S_R * ~SYNC) + (((SL.OUT.501H * SL.IN.501H) + (~SL.OUT.501H * ~SL.IN.501H)) * VCOR * SYNC.WAIT)	TO	501H.S_R;
ASSIGN	SL.OUT.501H * SYNC * (NORMAL + SL.IN.501H + ~SL.IN.SYNC + ~VCOR)	TO	501H;

ASSIGN	611R * SL.OUT.01RWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * SL.OUT.05RWCR * ~SL.OUT.05LS * SL.OUT.08RWCR * ~SL.OUT.08LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 05TPS * ~05TE * 06TPS * ~06TE * 06ES * 07TPS * 12TPS * ~SL.OUT.611_612AS * ~611_612TE * SL.OUT.460AS * ~460TE	TO	SL.OUT.611H;
ASSIGN	(611H.S_R * ~SYNC) + (((SL.OUT.611H * SL.IN.611H) + (~SL.OUT.611H * ~SL.IN.611H)) * VCOR * SYNC.WAIT)	TO	611H.S_R;
ASSIGN	SL.OUT.611H * SYNC * (NORMAL + SL.IN.611H + ~SL.IN.SYNC + ~VCOR)	TO	611H;
ASSIGN	612R * SL.OUT.01RWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * SL.OUT.05RWCR * ~SL.OUT.05LS * SL.OUT.08NWCR * ~SL.OUT.08LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 05TPS * ~05TE * 06TPS * ~06TE * 06ES * 07TPS * 12TPS * ~SL.OUT.611_612AS * ~611_612TE * SL.OUT.460AS * ~460TE	TO	SL.OUT.612H;
ASSIGN	(612H.S_R * ~SYNC) + (((SL.OUT.612H * SL.IN.612H) + (~SL.OUT.612H * ~SL.IN.612H)) * VCOR * SYNC.WAIT)	TO	612H.S_R;
ASSIGN	SL.OUT.612H * SYNC * (NORMAL + SL.IN.612H + ~SL.IN.SYNC + ~VCOR)	TO	612H;
ASSIGN	432R * SL.OUT.01RWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * SL.OUT.05NWCR * ~SL.OUT.05LS * SL.OUT.07RWCR * ~SL.OUT.07LS * SL.OUT.11NWCR * ~SL.OUT.11LS * SL.OUT.12RWCR * ~SL.OUT.12LS	TO	432H1;
ASSIGN	432H1 * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 05TPS * ~05TE * 05ES * 06TPS * ~06TE * 07TPS * ~07TE * ~07WS * 08TPS * ~08TE * 09TPS * ~09TE * 10TPS * ~10TE * 10ES * 11TPS * ~11TE * 12TPS * ~SL.OUT.432_534AS * ~432_531_534TE * SL.OUT.362AS * ~331_431_362TE * SL.OUT.460AS * ~460TE	TO	SL.OUT.432H;
ASSIGN	(432H.S_R * ~SYNC) + (((SL.OUT.432H * SL.IN.432H) + (~SL.OUT.432H * ~SL.IN.432H)) * VCOR * SYNC.WAIT)	TO	432H.S_R;
ASSIGN	SL.OUT.432H * SYNC * (NORMAL + SL.IN.432H + ~SL.IN.SYNC + ~VCOR)	TO	432H;
ASSIGN	431R * SL.OUT.01RWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS *	TO	SL.OUT.431H;

	SL.OUT.03NWCR * ~SL.OUT.03LS * SL.OUT.05NWCR * ~SL.OUT.05LS * SL.OUT.07NWCR * ~SL.OUT.07LS * SL.OUT.10RWCR * ~SL.OUT.10LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 05TPS * ~05TE * 05ES * 06TPS * ~06TE * 07TPS * ~07TE * ~07WS * 08TPS * ~08TE * 11TPS * ~11TE * 11ES * 12TPS * ~SL.OUT.331_431AS * ~331_431_362TE * SL.OUT.362AS * SL.OUT.460AS * ~460TE		
ASSIGN	(431H.S_R * ~SYNC) + (((SL.OUT.431H * SL.IN.431H) + (~SL.OUT.431H * ~SL.IN.431H)) * VCOR * SYNC.WAIT)	TO	431H.S_R;
ASSIGN	SL.OUT.431H * SYNC * (NORMAL + SL.IN.431H + ~SL.IN.SYNC + ~VCOR)	TO	431H;
ASSIGN	312R * ~SL.OUT.03LS * (SL.OUT.03NWCR * SL.OUT.340AS * ~16WS + SL.OUT.03RWCR * SL.OUT.02NWCR * ~SL.OUT.02LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * SL.OUT.460AS * ~460TE) * 16TPS * ~16TE * 16ES * ~SL.OUT.312AS * ~312_340TE	TO	SL.OUT.312H;
ASSIGN	(312H.S_R * ~SYNC) + (((SL.OUT.312H * SL.IN.312H) + (~SL.OUT.312H * ~SL.IN.312H)) * VCOR * SYNC.WAIT)	TO	312H.S_R;
ASSIGN	SL.OUT.312H * SYNC * (NORMAL + SL.IN.312H + ~SL.IN.SYNC + ~VCOR)	TO	312H;
ASSIGN	311R * SL.OUT.01NWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 04ES * 05TPS * ~05TE * 12TPS * ~12TE * 12ES * ~12WS * ~SL.OUT.04LS * SL.OUT.04NWCR * ~SL.OUT.311AS * ~311_361TE * 07TPS * SL.OUT.361AS * SL.OUT.460AS * ~460TE	TO	SL.OUT.311H;
ASSIGN	(311H.S_R * ~SYNC) + (((SL.OUT.311H * SL.IN.311H) + (~SL.OUT.311H * ~SL.IN.311H)) * VCOR * SYNC.WAIT)	TO	311H.S_R;
ASSIGN	SL.OUT.311H * SYNC * (NORMAL + SL.IN.311H + ~SL.IN.SYNC + ~VCOR)	TO	311H;
ASSIGN	334R * SL.OUT.01NWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 04ES * 05TPS * ~05TE * 12TPS * ~12TE * ~12WS * ~SL.OUT.04LS * SL.OUT.04RWCR * ~SL.OUT.06LS * SL.OUT.06RWCR * 13TPS * 13ES * 14TPS * ~14TE * ~SL.OUT.334AS * ~332_334TE * 07TPS * SL.OUT.361AS * ~311_361TE * SL.OUT.460AS * ~460TE	TO	SL.OUT.334H;

ASSIGN	(334H.S_R * ~SYNC) + (((SL.OUT.334H * SL.IN.334H) + (~SL.OUT.334H * ~SL.IN.334H)) * VCOR * SYNC.WAIT)	TO	334H.S_R;
ASSIGN	SL.OUT.334H * SYNC * (NORMAL + SL.IN.334H + ~SL.IN.SYNC + ~VCOR)	TO	334H;
ASSIGN	333R * SL.OUT.01NWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 04ES * 05TPS * ~05TE * 12TPS * ~12TE * ~12WS * ~SL.OUT.04LS * SL.OUT.04RWCR * ~SL.OUT.06LS * SL.OUT.06NWCR * ~SL.OUT.09LS * SL.OUT.09RWCR * 13TPS * 14TPS * ~14TE * 14ES * ~SL.OUT.332_333AS * ~332_334TE * 07TPS * SL.OUT.361AS * ~311_361TE * SL.OUT.460AS * ~460TE	TO	SL.OUT.333H;
ASSIGN	(333H.S_R * ~SYNC) + (((SL.OUT.333H * SL.IN.333H) + (~SL.OUT.333H * ~SL.IN.333H)) * VCOR * SYNC.WAIT)	TO	333H.S_R;
ASSIGN	SL.OUT.333H * SYNC * (NORMAL + SL.IN.333H + ~SL.IN.SYNC + ~VCOR)	TO	333H;
ASSIGN	332R * SL.OUT.01NWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 04ES * 05TPS * ~05TE * 12TPS * ~12TE * ~12WS * ~SL.OUT.04LS * SL.OUT.04RWCR * ~SL.OUT.06LS * SL.OUT.06NWCR * ~SL.OUT.09LS * SL.OUT.09NWCR * 13TPS * 14TPS * ~14TE * 14ES * ~SL.OUT.332_333AS * ~332_334TE * 07TPS * SL.OUT.361AS * ~311_361TE * SL.OUT.460AS * ~460TE	TO	SL.OUT.332H;
ASSIGN	(332H.S_R * ~SYNC) + (((SL.OUT.332H * SL.IN.332H) + (~SL.OUT.332H * ~SL.IN.332H)) * VCOR * SYNC.WAIT)	TO	332H.S_R;
ASSIGN	SL.OUT.332H * SYNC * (NORMAL + SL.IN.332H + ~SL.IN.SYNC + ~VCOR)	TO	332H;
ASSIGN	331R * SL.OUT.01RWCR * ~SL.OUT.01LS * SL.OUT.02NWCR * ~SL.OUT.02LS * SL.OUT.03NWCR * ~SL.OUT.03LS * SL.OUT.05NWCR * ~SL.OUT.05LS * SL.OUT.07NWCR * ~SL.OUT.07LS * SL.OUT.10NWCR * ~SL.OUT.10LS * 03TPS * ~03TE * ~03WS * 04TPS * ~04TE * 05TPS * ~05TE * 05ES * 06TPS * ~06TE * 07TPS * ~07TE * ~07WS * 08TPS * ~08TE * 11TPS * ~11TE * 11ES * 12TPS * ~SL.OUT.331_431AS * ~331_431_362TE * SL.OUT.362AS * SL.OUT.460AS * ~460TE	TO	SL.OUT.331H;

ASSIGN	(331H.S_R * ~SYNC) + (((SL.OUT.331H * SL.IN.331H) + (~SL.OUT.331H * ~SL.IN.331H)) * VCOR * SYNC.WAIT)	TO	331H.S_R;
ASSIGN	SL.OUT.331H * SYNC * (NORMAL + SL.IN.331H + ~SL.IN.SYNC + ~VCOR)	TO	331H;
ASSIGN	361R * SL.OUT.04NWCR * ~SL.OUT.04LS * 12TPS * ~12TE * 12WS * SL.OUT.15TS * ~SL.OUT.15ES * 15WS * ~SL.OUT.361AS * ~311_361TE * SL.OUT.311AS	TO	SL.OUT.361H;
ASSIGN	(361H.S_R * ~SYNC) + (((SL.OUT.361H * SL.IN.361H) + (~SL.OUT.361H * ~SL.IN.361H)) * VCOR * SYNC.WAIT)	TO	361H.S_R;
ASSIGN	SL.OUT.361H * SYNC * (NORMAL + SL.IN.361H + ~SL.IN.SYNC + ~VCOR)	TO	361H;
ASSIGN	361R * ~SL.OUT.04NWCR * SL.OUT.04RWCR * ~SL.OUT.04LS * 12TPS * ~12TE * 12WS * 13TPS * 14TPS * ~14TE * ~SL.OUT.06LS * (~14ES * SL.OUT.06NWCR * (SL.OUT.09RWCR + SL.OUT.09NWCR) * ~SL.OUT.09LS * SL.OUT.332_333AS + SL.OUT.06RWCR * SL.OUT.334AS * ~13ES) * ~SL.OUT.361AS * ~311_361TE * ~332_334TE	TO	SL.OUT.361L;
ASSIGN	(361L.S_R * ~SYNC) + (((SL.OUT.361L * SL.IN.361L) + (~SL.OUT.361L * ~SL.IN.361L)) * VCOR * SYNC.WAIT)	TO	361L.S_R;
ASSIGN	SL.OUT.361L * SYNC * (NORMAL + SL.IN.361L + ~SL.IN.SYNC + ~VCOR)	TO	361L;
ASSIGN	362R * 07TPS * ~07TE * 07WS * 08TPS * ~08TE * ~SL.OUT.07LS * 11TPS * ~11TE * (SL.OUT.07NWCR * SL.OUT.10RWCR * ~SL.OUT.10LS * 21TPS * ~SL.OUT.21ES * 21WS * SL.OUT.331_431AS *~21TE + SL.OUT.07RWCR * SL.OUT.11NWCR * ~SL.OUT.11LS * SL.OUT.12RWCR * ~SL.OUT.12LS * 09TPS * ~09TE * 10TPS * ~10TE * 20TS * ~SL.OUT.20ES * 20WS * SL.OUT.432_534AS * ~432_531_534TE) * ~SL.OUT.362AS * ~331_431_362TE	TO	SL.OUT.362H;
ASSIGN	(362H.S_R * ~SYNC) + (((SL.OUT.362H * SL.IN.362H) + (~SL.OUT.362H * ~SL.IN.362H)) * VCOR * SYNC.WAIT)	TO	362H.S_R;
ASSIGN	SL.OUT.362H * SYNC * (NORMAL + SL.IN.362H + ~SL.IN.SYNC + ~VCOR)	TO	362H;
ASSIGN	362R * ~SL.OUT.07LS * 11TPS *~11TE * 08TPS * ~08TE * 07TPS * ~07TE * 07WS * ~SL.OUT.362AS * ~331_431_362TE * (~11ES * SL.OUT.10NWCR * ~SL.OUT.10LS * SL.OUT.07NWCR * SL.OUT.331_431AS +	TO	SL.OUT.362L;

	~SL.OUT.11LS * 10TPS * ~10TE * 09TPS * ~09TE * SL.OUT.07RWCR * ~432_531_534TE * (~10ES * SL.OUT.12NWCR * ~SL.OUT.12LS * SL.OUT.11NWCR * SL.OUT.432_534AS + ~09ES * SL.OUT.11RWCR * ~SL.OUT.13LS * (SL.OUT.13RWCR + SL.OUT.13NWCR * ~SL.OUT.14LS * (SL.OUT.14RWCR + SL.OUT.14NWCR)) * SL.OUT.531_532_533AS))		
ASSIGN	(362L.S_R * ~SYNC) + (((SL.OUT.362L * SL.IN.362L) + (~SL.OUT.362L * ~SL.IN.362L)) * VCOR * SYNC.WAIT)	TO	362L.S_R;
ASSIGN	SL.OUT.362L * SYNC * (NORMAL + SL.IN.362L + ~SL.IN.SYNC + ~VCOR)	TO	362L;
ASSIGN	340R * SL.OUT.03NWCR * ~SL.OUT.03LS * 16TPS * ~16TE * 16WS * SL.OUT.17TS * ~SL.OUT.17ES * 17WS * ~SL.OUT.340AS * ~312_340TE * SL.OUT.312AS	TO	SL.OUT.340H;
ASSIGN	(340H.S_R * ~SYNC) + (((SL.OUT.340H * SL.IN.340H) + (~SL.OUT.340H * ~SL.IN.340H)) * VCOR * SYNC.WAIT)	TO	340H.S_R;
ASSIGN	SL.OUT.340H * SYNC * (NORMAL + SL.IN.340H + ~SL.IN.SYNC + ~VCOR)	TO	340H;
ASSIGN	10TPS * ~10TE * 09TPS * ~09TE * ~432_531_534TE * SL.OUT.07RWCR * ~11TE * 11TPS * 08TPS * ~08TE * ~SL.OUT.07LS * ~331_431_362TE * 07TPS * ~07TE * SL.OUT.362AS * 05ES * ~07WS * SL.OUT.05NWCR * ~SL.OUT.05LS * 06TPS * ~06TE * SL.OUT.01RWCR * ~SL.OUT.01LS * 12TPS * 05TPS * ~05TE * SL.OUT.03NWCR * 04TPS * ~04TE * ~SL.OUT.03LS * SL.OUT.02NWCR * ~SL.OUT.02LS * ~460TE * 03TPS * ~03TE * SL.OUT.460AS * ~03WS * ~SL.OUT.11LS	TO	531H1, 532H1, 533H1, 534H1;
ASSIGN	531H1 * 531R * 09ES * SL.OUT.11RWCR * SL.OUT.13NWCR * ~SL.OUT.13LS * SL.OUT.14NWCR * ~SL.OUT.14LS * ~SL.OUT.531_532_533AS	TO	SL.OUT.531H;
ASSIGN	(531H.S_R * ~SYNC) + (((SL.OUT.531H * SL.IN.531H) + (~SL.OUT.531H * ~SL.IN.531H)) * VCOR * SYNC.WAIT)	TO	531H.S_R;
ASSIGN	SL.OUT.531H * SYNC * (NORMAL + SL.IN.531H + ~SL.IN.SYNC + ~VCOR)	TO	531H;
ASSIGN	532H1 * 532R * 09ES * SL.OUT.11RWCR * SL.OUT.13NWCR * ~SL.OUT.13LS * SL.OUT.14RWCR * ~SL.OUT.14LS * ~SL.OUT.531_532_533AS	TO	SL.OUT.532H;

ASSIGN	(532H.S_R * ~SYNC) + (((SL.OUT.532H * SL.IN.532H) + (~SL.OUT.532H * ~SL.IN.532H)) * VCOR * SYNC.WAIT)	TO	532H.S_R;
ASSIGN	SL.OUT.532H * SYNC * (NORMAL + SL.IN.532H + ~SL.IN.SYNC + ~VCOR)	TO	532H;
ASSIGN	533H1 * 533R * 09ES * SL.OUT.11RWCR * SL.OUT.13RWCR * ~SL.OUT.13LS * ~SL.OUT.531_532_533AS	TO	SL.OUT.533H;
ASSIGN	(533H.S_R * ~SYNC) + (((SL.OUT.533H * SL.IN.533H) + (~SL.OUT.533H * ~SL.IN.533H)) * VCOR * SYNC.WAIT)	TO	533H.S_R;
ASSIGN	SL.OUT.533H * SYNC * (NORMAL + SL.IN.533H + ~SL.IN.SYNC + ~VCOR)	TO	533H;
ASSIGN	534H1 * 534R * 10ES * SL.OUT.11NWCR * SL.OUT.12NWCR * ~SL.OUT.12LS * ~SL.OUT.432_534AS	TO	SL.OUT.534H;
ASSIGN	(534H.S_R * ~SYNC) + (((SL.OUT.534H * SL.IN.534H) + (~SL.OUT.534H * ~SL.IN.534H)) * VCOR * SYNC.WAIT)	TO	534H.S_R;
ASSIGN	SL.OUT.534H * SYNC * (NORMAL + SL.IN.534H + ~SL.IN.SYNC + ~VCOR)	TO	534H;

/* **RED SIGNAL CONTROL (DEMO ONLY)** */

ASSIGN	~SL.OUT.311H	TO	SL.OUT.311RK;
ASSIGN	(311RK.S_R * ~SYNC) + (((SL.OUT.311RK * SL.IN.311RK) + (~SL.OUT.311RK * ~SL.IN.311RK)) * VCOR * SYNC.WAIT)	TO	311RK.S_R;
ASSIGN	SL.OUT.311RK * SYNC * (NORMAL + SL.IN.311RK + ~SL.IN.SYNC + ~VCOR)	TO	311RK;
ASSIGN	~SL.OUT.312H	TO	SL.OUT.312RK;
ASSIGN	(312RK.S_R * ~SYNC) + (((SL.OUT.312RK * SL.IN.312RK) + (~SL.OUT.312RK * ~SL.IN.312RK)) * VCOR * SYNC.WAIT)	TO	312RK.S_R;
ASSIGN	SL.OUT.312RK * SYNC * (NORMAL + SL.IN.312RK + ~SL.IN.SYNC + ~VCOR)	TO	312RK;
ASSIGN	~SL.OUT.331H	TO	SL.OUT.331RK;

ASSIGN	(331RK.S_R * ~SYNC) + (((SL.OUT.331RK * SL.IN.331RK) + (~SL.OUT.331RK * ~SL.IN.331RK)) * VCOR * SYNC.WAIT)	TO	331RK.S_R;
ASSIGN	SL.OUT.331RK * SYNC * (NORMAL + SL.IN.331RK + ~SL.IN.SYNC + ~VCOR)	TO	331RK;
ASSIGN	~SL.OUT.332H	TO	SL.OUT.332RK;
ASSIGN	(332RK.S_R * ~SYNC) + (((SL.OUT.332RK * SL.IN.332RK) + (~SL.OUT.332RK * ~SL.IN.332RK)) * VCOR * SYNC.WAIT)	TO	332RK.S_R;
ASSIGN	SL.OUT.332RK * SYNC * (NORMAL + SL.IN.332RK + ~SL.IN.SYNC + ~VCOR)	TO	332RK;
ASSIGN	~SL.OUT.333H	TO	SL.OUT.333RK;
ASSIGN	(333RK.S_R * ~SYNC) + (((SL.OUT.333RK * SL.IN.333RK) + (~SL.OUT.333RK * ~SL.IN.333RK)) * VCOR * SYNC.WAIT)	TO	333RK.S_R;
ASSIGN	SL.OUT.333RK * SYNC * (NORMAL + SL.IN.333RK + ~SL.IN.SYNC + ~VCOR)	TO	333RK;
ASSIGN	~SL.OUT.334H	TO	SL.OUT.334RK;
ASSIGN	(334RK.S_R * ~SYNC) + (((SL.OUT.334RK * SL.IN.334RK) + (~SL.OUT.334RK * ~SL.IN.334RK)) * VCOR * SYNC.WAIT)	TO	334RK.S_R;
ASSIGN	SL.OUT.334RK * SYNC * (NORMAL + SL.IN.334RK + ~SL.IN.SYNC + ~VCOR)	TO	334RK;
ASSIGN	~SL.OUT.340H	TO	SL.OUT.340RK;
ASSIGN	(340RK.S_R * ~SYNC) + (((SL.OUT.340RK * SL.IN.340RK) + (~SL.OUT.340RK * ~SL.IN.340RK)) * VCOR * SYNC.WAIT)	TO	340RK.S_R;
ASSIGN	SL.OUT.340RK * SYNC * (NORMAL + SL.IN.340RK + ~SL.IN.SYNC + ~VCOR)	TO	340RK;
ASSIGN	~SL.OUT.361H * ~SL.OUT.361L	TO	SL.OUT.361RK;
ASSIGN	(361RK.S_R * ~SYNC) + (((SL.OUT.361RK * SL.IN.361RK) + (~SL.OUT.361RK * ~SL.IN.361RK)) * VCOR * SYNC.WAIT)	TO	361RK.S_R;
ASSIGN	SL.OUT.361RK * SYNC * (NORMAL + SL.IN.361RK + ~SL.IN.SYNC + ~VCOR)	TO	361RK;

ASSIGN	~SL.OUT.362H * ~SL.OUT.362L	TO	SL.OUT.362RK;
ASSIGN	(362RK.S_R * ~SYNC) + (((SL.OUT.362RK * SL.IN.362RK) + (~SL.OUT.362RK * ~SL.IN.362RK)) * VCOR * SYNC.WAIT)	TO	362RK.S_R;
ASSIGN	SL.OUT.362RK * SYNC * (NORMAL + SL.IN.362RK + ~SL.IN.SYNC + ~VCOR)	TO	362RK;
ASSIGN	~SL.OUT.431H	TO	SL.OUT.431RK;
ASSIGN	(431RK.S_R * ~SYNC) + (((SL.OUT.431RK * SL.IN.431RK) + (~SL.OUT.431RK * ~SL.IN.431RK)) * VCOR * SYNC.WAIT)	TO	431RK.S_R;
ASSIGN	SL.OUT.431RK * SYNC * (NORMAL + SL.IN.431RK + ~SL.IN.SYNC + ~VCOR)	TO	431RK;
ASSIGN	~SL.OUT.432H	TO	SL.OUT.432RK;
ASSIGN	(432RK.S_R * ~SYNC) + (((SL.OUT.432RK * SL.IN.432RK) + (~SL.OUT.432RK * ~SL.IN.432RK)) * VCOR * SYNC.WAIT)	TO	432RK.S_R;
ASSIGN	SL.OUT.432RK * SYNC * (NORMAL + SL.IN.432RK + ~SL.IN.SYNC + ~VCOR)	TO	432RK;
ASSIGN	~SL.OUT.460H * ~SL.OUT.460L	TO	SL.OUT.460RK;
ASSIGN	(460RK.S_R * ~SYNC) + (((SL.OUT.460RK * SL.IN.460RK) + (~SL.OUT.460RK * ~SL.IN.460RK)) * VCOR * SYNC.WAIT)	TO	460RK.S_R;
ASSIGN	SL.OUT.460RK * SYNC * (NORMAL + SL.IN.460RK + ~SL.IN.SYNC + ~VCOR)	TO	460RK;
ASSIGN	~SL.OUT.501H	TO	SL.OUT.501RK;
ASSIGN	(501RK.S_R * ~SYNC) + (((SL.OUT.501RK * SL.IN.501RK) + (~SL.OUT.501RK * ~SL.IN.501RK)) * VCOR * SYNC.WAIT)	TO	501RK.S_R;
ASSIGN	SL.OUT.501RK * SYNC * (NORMAL + SL.IN.501RK + ~SL.IN.SYNC + ~VCOR)	TO	501RK;
ASSIGN	~SL.OUT.512H	TO	SL.OUT.512RK;
ASSIGN	(512RK.S_R * ~SYNC) + (((SL.OUT.512RK * SL.IN.512RK) + (~SL.OUT.512RK * ~SL.IN.512RK)) * VCOR * SYNC.WAIT)	TO	512RK.S_R;

ASSIGN	SL.OUT.512RK * SYNC * (NORMAL + SL.IN.512RK + ~SL.IN.SYNC + ~VCOR)	TO	512RK;
ASSIGN	~SL.OUT.531H	TO	SL.OUT.531RK;
ASSIGN	(531RK.S_R * ~SYNC) + (((SL.OUT.531RK * SL.IN.531RK) + (~SL.OUT.531RK * ~SL.IN.531RK)) * VCOR * SYNC.WAIT)	TO	531RK.S_R;
ASSIGN	SL.OUT.531RK * SYNC * (NORMAL + SL.IN.531RK + ~SL.IN.SYNC + ~VCOR)	TO	531RK;
ASSIGN	~SL.OUT.532H	TO	SL.OUT.532RK;
ASSIGN	(532RK.S_R * ~SYNC) + (((SL.OUT.532RK * SL.IN.532RK) + (~SL.OUT.532RK * ~SL.IN.532RK)) * VCOR * SYNC.WAIT)	TO	532RK.S_R;
ASSIGN	SL.OUT.532RK * SYNC * (NORMAL + SL.IN.532RK + ~SL.IN.SYNC + ~VCOR)	TO	532RK;
ASSIGN	~SL.OUT.533H	TO	SL.OUT.533RK;
ASSIGN	(533RK.S_R * ~SYNC) + (((SL.OUT.533RK * SL.IN.533RK) + (~SL.OUT.533RK * ~SL.IN.533RK)) * VCOR * SYNC.WAIT)	TO	533RK.S_R;
ASSIGN	SL.OUT.533RK * SYNC * (NORMAL + SL.IN.533RK + ~SL.IN.SYNC + ~VCOR)	TO	533RK;
ASSIGN	~SL.OUT.534H	TO	SL.OUT.534RK;
ASSIGN	(534RK.S_R * ~SYNC) + (((SL.OUT.534RK * SL.IN.534RK) + (~SL.OUT.534RK * ~SL.IN.534RK)) * VCOR * SYNC.WAIT)	TO	534RK.S_R;
ASSIGN	SL.OUT.534RK * SYNC * (NORMAL + SL.IN.534RK + ~SL.IN.SYNC + ~VCOR)	TO	534RK;
ASSIGN	~SL.OUT.540H	TO	SL.OUT.540RK;
ASSIGN	(540RK.S_R * ~SYNC) + (((SL.OUT.540RK * SL.IN.540RK) + (~SL.OUT.540RK * ~SL.IN.540RK)) * VCOR * SYNC.WAIT)	TO	540RK.S_R;
ASSIGN	SL.OUT.540RK * SYNC * (NORMAL + SL.IN.540RK + ~SL.IN.SYNC + ~VCOR)	TO	540RK;
ASSIGN	~SL.OUT.611H	TO	SL.OUT.611RK;

ASSIGN	(611RK.S_R * ~SYNC) + (((SL.OUT.611RK * SL.IN.611RK) + (~SL.OUT.611RK * ~SL.IN.611RK)) * VCOR * SYNC.WAIT)	TO	611RK.S_R;
ASSIGN	SL.OUT.611RK * SYNC * (NORMAL + SL.IN.611RK + ~SL.IN.SYNC + ~VCOR)	TO	611RK;
ASSIGN	~SL.OUT.612H	TO	SL.OUT.612RK;
ASSIGN	(612RK.S_R * ~SYNC) + (((SL.OUT.612RK * SL.IN.612RK) + (~SL.OUT.612RK * ~SL.IN.612RK)) * VCOR * SYNC.WAIT)	TO	612RK.S_R;
ASSIGN	SL.OUT.612RK * SYNC * (NORMAL + SL.IN.612RK + ~SL.IN.SYNC + ~VCOR)	TO	612RK;

/* **BITS TRANSFERRED FROM NON VITAL PROCESSOR** */

ASSIGN	(311R.NNV * N.NV.HEALTH) + (311R.SNV * ~N.NV.HEALTH)	TO	311R;
ASSIGN	(312R.NNV * N.NV.HEALTH) + (312R.SNV * ~N.NV.HEALTH)	TO	312R;
ASSIGN	(331R.NNV * N.NV.HEALTH) + (331R.SNV * ~N.NV.HEALTH)	TO	331R;
ASSIGN	(332R.NNV * N.NV.HEALTH) + (332R.SNV * ~N.NV.HEALTH)	TO	332R;
ASSIGN	(333R.NNV * N.NV.HEALTH) + (333R.SNV * ~N.NV.HEALTH)	TO	333R;
ASSIGN	(334R.NNV * N.NV.HEALTH) + (334R.SNV * ~N.NV.HEALTH)	TO	334R;
ASSIGN	(340R.NNV * N.NV.HEALTH) + (340R.SNV * ~N.NV.HEALTH)	TO	340R;
ASSIGN	(361R.NNV * N.NV.HEALTH) + (361R.SNV * ~N.NV.HEALTH)	TO	361R;
ASSIGN	(362R.NNV * N.NV.HEALTH) + (362R.SNV * ~N.NV.HEALTH)	TO	362R;
ASSIGN	(431R.NNV * N.NV.HEALTH) + (431R.SNV * ~N.NV.HEALTH)	TO	431R;
ASSIGN	(432R.NNV * N.NV.HEALTH) + (432R.SNV * ~N.NV.HEALTH)	TO	432R;
ASSIGN	(460R.NNV * N.NV.HEALTH) + (460R.SNV * ~N.NV.HEALTH)	TO	460R;
ASSIGN	(501R.NNV * N.NV.HEALTH) + (501R.SNV * ~N.NV.HEALTH)	TO	501R;

ASSIGN	(512R.NNV * N.NV.HEALTH) + (512R.SNV * ~N.NV.HEALTH)	TO	512R;
ASSIGN	(531R.NNV * N.NV.HEALTH) + (531R.SNV * ~N.NV.HEALTH)	TO	531R;
ASSIGN	(532R.NNV * N.NV.HEALTH) + (532R.SNV * ~N.NV.HEALTH)	TO	532R;
ASSIGN	(533R.NNV * N.NV.HEALTH) + (533R.SNV * ~N.NV.HEALTH)	TO	533R;
ASSIGN	(534R.NNV * N.NV.HEALTH) + (534R.SNV * ~N.NV.HEALTH)	TO	534R;
ASSIGN	(540R.NNV * N.NV.HEALTH) + (540R.SNV * ~N.NV.HEALTH)	TO	540R;
ASSIGN	(611R.NNV * N.NV.HEALTH) + (611R.SNV * ~N.NV.HEALTH)	TO	611R;
ASSIGN	(612R.NNV * N.NV.HEALTH) + (612R.SNV * ~N.NV.HEALTH)	TO	612R;
ASSIGN	(01NLP.NNV * N.NV.HEALTH) + (01NLP.SNV * ~N.NV.HEALTH)	TO	01NLP;
ASSIGN	(01RLP.NNV * N.NV.HEALTH) + (01RLP.SNV * ~N.NV.HEALTH)	TO	01RLP;
ASSIGN	(02NLP.NNV * N.NV.HEALTH) + (02NLP.SNV * ~N.NV.HEALTH)	TO	02NLP;
ASSIGN	(02RLP.NNV * N.NV.HEALTH) + (02RLP.SNV * ~N.NV.HEALTH)	TO	02RLP;
ASSIGN	(03NLP.NNV * N.NV.HEALTH) + (03NLP.SNV * ~N.NV.HEALTH)	TO	03NLP;
ASSIGN	(03RLP.NNV * N.NV.HEALTH) + (03RLP.SNV * ~N.NV.HEALTH)	TO	03RLP;
ASSIGN	(04NLP.NNV * N.NV.HEALTH) + (04NLP.SNV * ~N.NV.HEALTH)	TO	04NLP;
ASSIGN	(04RLP.NNV * N.NV.HEALTH) + (04RLP.SNV * ~N.NV.HEALTH)	TO	04RLP;
ASSIGN	(05NLP.NNV * N.NV.HEALTH) + (05NLP.SNV * ~N.NV.HEALTH)	TO	05NLP;
ASSIGN	(05RLP.NNV * N.NV.HEALTH) + (05RLP.SNV * ~N.NV.HEALTH)	TO	05RLP;
ASSIGN	(06NLP.NNV * N.NV.HEALTH) + (06NLP.SNV * ~N.NV.HEALTH)	TO	06NLP;
ASSIGN	(06RLP.NNV * N.NV.HEALTH) + (06RLP.SNV * ~N.NV.HEALTH)	TO	06RLP;
ASSIGN	(07NLP.NNV * N.NV.HEALTH) + (07NLP.SNV * ~N.NV.HEALTH)	TO	07NLP;

ASSIGN	(07RLP.NNV * N.NV.HEALTH) + (07RLP.SNV * ~N.NV.HEALTH)	TO	07RLP;
ASSIGN	(08NLP.NNV * N.NV.HEALTH) + (08NLP.SNV * ~N.NV.HEALTH)	TO	08NLP;
ASSIGN	(08RLP.NNV * N.NV.HEALTH) + (08RLP.SNV * ~N.NV.HEALTH)	TO	08RLP;
ASSIGN	(09NLP.NNV * N.NV.HEALTH) + (09NLP.SNV * ~N.NV.HEALTH)	TO	09NLP;
ASSIGN	(09RLP.NNV * N.NV.HEALTH) + (09RLP.SNV * ~N.NV.HEALTH)	TO	09RLP;
ASSIGN	(10NLP.NNV * N.NV.HEALTH) + (10NLP.SNV * ~N.NV.HEALTH)	TO	10NLP;
ASSIGN	(10RLP.NNV * N.NV.HEALTH) + (10RLP.SNV * ~N.NV.HEALTH)	TO	10RLP;
ASSIGN	(11NLP.NNV * N.NV.HEALTH) + (11NLP.SNV * ~N.NV.HEALTH)	TO	11NLP;
ASSIGN	(11RLP.NNV * N.NV.HEALTH) + (11RLP.SNV * ~N.NV.HEALTH)	TO	11RLP;
ASSIGN	(12NLP.NNV * N.NV.HEALTH) + (12NLP.SNV * ~N.NV.HEALTH)	TO	12NLP;
ASSIGN	(12RLP.NNV * N.NV.HEALTH) + (12RLP.SNV * ~N.NV.HEALTH)	TO	12RLP;
ASSIGN	(13NLP.NNV * N.NV.HEALTH) + (13NLP.SNV * ~N.NV.HEALTH)	TO	13NLP;
ASSIGN	(13RLP.NNV * N.NV.HEALTH) + (13RLP.SNV * ~N.NV.HEALTH)	TO	13RLP;
ASSIGN	(14NLP.NNV * N.NV.HEALTH) + (14NLP.SNV * ~N.NV.HEALTH)	TO	14NLP;
ASSIGN	(14RLP.NNV * N.NV.HEALTH) + (14RLP.SNV * ~N.NV.HEALTH)	TO	14RLP;

/* **BITS TRANSFERRED TO NON VITAL PROCESSOR** */

/* **SWITCH CORRESPONDENCE** */

NV.ASSIGN	01NWCR * SYNC	TO	01NWCR.NNV, 01NWCR.SNV;
NV.ASSIGN	01RWCR * SYNC	TO	01RWCR.NNV, 01RWCR.SNV;
NV.ASSIGN	02NWCR * SYNC	TO	02NWCR.NNV, 02NWCR.SNV;

NV.ASSIGN	02RWCR * SYNC	TO	02RWCR.NNV, 02RWCR.SNV;
NV.ASSIGN	03NWCR * SYNC	TO	03NWCR.NNV, 03NWCR.SNV;
NV.ASSIGN	03RWCR * SYNC	TO	03RWCR.NNV, 03RWCR.SNV;
NV.ASSIGN	04NWCR * SYNC	TO	04NWCR.NNV, 04NWCR.SNV;
NV.ASSIGN	04RWCR * SYNC	TO	04RWCR.NNV, 04RWCR.SNV;
NV.ASSIGN	05NWCR * SYNC	TO	05NWCR.NNV, 05NWCR.SNV;
NV.ASSIGN	05RWCR * SYNC	TO	05RWCR.NNV, 05RWCR.SNV;
NV.ASSIGN	06NWCR * SYNC	TO	06NWCR.NNV, 06NWCR.SNV;
NV.ASSIGN	06RWCR * SYNC	TO	06RWCR.NNV, 06RWCR.SNV;
NV.ASSIGN	07NWCR * SYNC	TO	07NWCR.NNV, 07NWCR.SNV;
NV.ASSIGN	07RWCR * SYNC	TO	07RWCR.NNV, 07RWCR.SNV;
NV.ASSIGN	08NWCR * SYNC	TO	08NWCR.NNV, 08NWCR.SNV;
NV.ASSIGN	08RWCR * SYNC	TO	08RWCR.NNV, 08RWCR.SNV;
NV.ASSIGN	09NWCR * SYNC	TO	09NWCR.NNV, 09NWCR.SNV;
NV.ASSIGN	09RWCR * SYNC	TO	09RWCR.NNV, 09RWCR.SNV;
NV.ASSIGN	10NWCR * SYNC	TO	10NWCR.NNV, 10NWCR.SNV;
NV.ASSIGN	10RWCR * SYNC	TO	10RWCR.NNV, 10RWCR.SNV;
NV.ASSIGN	11NWCR * SYNC	TO	11NWCR.NNV, 11NWCR.SNV;
NV.ASSIGN	11RWCR * SYNC	TO	11RWCR.NNV, 11RWCR.SNV;
NV.ASSIGN	12NWCR * SYNC	TO	12NWCR.NNV, 12NWCR.SNV;
NV.ASSIGN	12RWCR * SYNC	TO	12RWCR.NNV, 12RWCR.SNV;

NV.ASSIGN	13NWCR * SYNC	TO	13NWCR.NNV, 13NWCR.SNV;
NV.ASSIGN	13RWCR * SYNC	TO	13RWCR.NNV, 13RWCR.SNV;
NV.ASSIGN	14NWCR * SYNC	TO	14NWCR.NNV, 14NWCR.SNV;
NV.ASSIGN	14RWCR * SYNC	TO	14RWCR.NNV, 14RWCR.SNV;

/* **LOCK STICKS** */

NV.ASSIGN	01LS * SYNC	TO	01LS.NNV, 01LS.SNV;
NV.ASSIGN	02LS * SYNC	TO	02LS.NNV, 02LS.SNV;
NV.ASSIGN	03LS * SYNC	TO	03LS.NNV, 03LS.SNV;
NV.ASSIGN	04LS * SYNC	TO	04LS.NNV, 04LS.SNV;
NV.ASSIGN	05LS * SYNC	TO	05LS.NNV, 05LS.SNV;
NV.ASSIGN	06LS * SYNC	TO	06LS.NNV, 06LS.SNV;
NV.ASSIGN	07LS * SYNC	TO	07LS.NNV, 07LS.SNV;
NV.ASSIGN	08LS * SYNC	TO	08LS.NNV, 08LS.SNV;
NV.ASSIGN	09LS * SYNC	TO	09LS.NNV, 09LS.SNV;
NV.ASSIGN	10LS * SYNC	TO	10LS.NNV, 10LS.SNV;
NV.ASSIGN	11LS * SYNC	TO	11LS.NNV, 11LS.SNV;
NV.ASSIGN	12LS * SYNC	TO	12LS.NNV, 12LS.SNV;
NV.ASSIGN	13LS * SYNC	TO	13LS.NNV, 13LS.SNV;
NV.ASSIGN	14LS * SYNC	TO	14LS.NNV, 14LS.SNV;

/* **TRACK REPEATER STICKS** */

NV.ASSIGN	01TPS * SYNC	TO	01TPS.NNV, 01TPS.SNV;
NV.ASSIGN	02TPS * SYNC	TO	02TPS.NNV, 02TPS.SNV;
NV.ASSIGN	03TPS * SYNC	TO	03TPS.NNV, 03TPS.SNV;
NV.ASSIGN	04TPS * SYNC	TO	04TPS.NNV, 04TPS.SNV;
NV.ASSIGN	05TPS * SYNC	TO	05TPS.NNV, 05TPS.SNV;
NV.ASSIGN	06TPS * SYNC	TO	06TPS.NNV, 06TPS.SNV;
NV.ASSIGN	07TPS * SYNC	TO	07TPS.NNV, 07TPS.SNV;
NV.ASSIGN	08TPS * SYNC	TO	08TPS.NNV, 08TPS.SNV;
NV.ASSIGN	09TPS * SYNC	TO	09TPS.NNV, 09TPS.SNV;
NV.ASSIGN	10TPS * SYNC	TO	10TPS.NNV, 10TPS.SNV;
NV.ASSIGN	11TPS * SYNC	TO	11TPS.NNV, 11TPS.SNV;
NV.ASSIGN	12TPS * SYNC	TO	12TPS.NNV, 12TPS.SNV;
NV.ASSIGN	13TPS * SYNC	TO	13TPS.NNV, 13TPS.SNV;
NV.ASSIGN	14TPS * SYNC	TO	14TPS.NNV, 14TPS.SNV;
NV.ASSIGN	15TPS * SYNC	TO	15TPS.NNV, 15TPS.SNV;
NV.ASSIGN	16TPS * SYNC	TO	16TPS.NNV,

			16TPS.SNV;
NV.ASSIGN	17TPS * SYNC	TO	17TPS.NNV, 17TPS.SNV;
NV.ASSIGN	20TPS * SYNC	TO	20TPS.NNV, 20TPS.SNV;
NV.ASSIGN	21TPS * SYNC	TO	21TPS.NNV, 21TPS.SNV;
NV.ASSIGN	26TPS * SYNC	TO	26TPS.NNV, 26TPS.SNV;
NV.ASSIGN	35TPS * SYNC	TO	35TPS.NNV, 35TPS.SNV;

/* **DIRECTIONAL ROUTE STICKS** */

NV.ASSIGN	01ES * SYNC	TO	01ES.NNV, 01ES.SNV;
NV.ASSIGN	01WS * SYNC	TO	01WS.NNV, 01WS.SNV;
NV.ASSIGN	02ES * SYNC	TO	02ES.NNV, 02ES.SNV;
NV.ASSIGN	02WS * SYNC	TO	02WS.NNV, 02WS.SNV;
NV.ASSIGN	03ES * SYNC	TO	03ES.NNV, 03ES.SNV;
NV.ASSIGN	03WS * SYNC	TO	03WS.NNV, 03WS.SNV;
NV.ASSIGN	04ES * SYNC	TO	04ES.NNV, 04ES.SNV;
NV.ASSIGN	04WS * SYNC	TO	04WS.NNV, 04WS.SNV;
NV.ASSIGN	05ES * SYNC	TO	05ES.NNV, 05ES.SNV;
NV.ASSIGN	05WS * SYNC	TO	05WS.NNV, 05WS.SNV;
NV.ASSIGN	06ES * SYNC	TO	06ES.NNV, 06ES.SNV;
NV.ASSIGN	06WS * SYNC	TO	06WS.NNV, 06WS.SNV;

NV.ASSIGN	07ES * SYNC	TO	07ES.NNV, 07ES.SNV;
NV.ASSIGN	07WS * SYNC	TO	07WS.NNV, 07WS.SNV;
NV.ASSIGN	08ES * SYNC	TO	08ES.NNV, 08ES.SNV;
NV.ASSIGN	08WS * SYNC	TO	08WS.NNV, 08WS.SNV;
NV.ASSIGN	09ES * SYNC	TO	09ES.NNV, 09ES.SNV;
NV.ASSIGN	09WS * SYNC	TO	09WS.NNV, 09WS.SNV;
NV.ASSIGN	10ES * SYNC	TO	10ES.NNV, 10ES.SNV;
NV.ASSIGN	10WS * SYNC	TO	10WS.NNV, 10WS.SNV;
NV.ASSIGN	11ES * SYNC	TO	11ES.NNV, 11ES.SNV;
NV.ASSIGN	11WS * SYNC	TO	11WS.NNV, 11WS.SNV;
NV.ASSIGN	12ES * SYNC	TO	12ES.NNV, 12ES.SNV;
NV.ASSIGN	12WS * SYNC	TO	12WS.NNV, 12WS.SNV;
NV.ASSIGN	13ES * SYNC	TO	13ES.NNV, 13ES.SNV;
NV.ASSIGN	13WS * SYNC	TO	13WS.NNV, 13WS.SNV;
NV.ASSIGN	14ES * SYNC	TO	14ES.NNV, 14ES.SNV;
NV.ASSIGN	14WS * SYNC	TO	14WS.NNV, 14WS.SNV;
NV.ASSIGN	15ES * SYNC	TO	15ES.NNV, 15ES.SNV;
NV.ASSIGN	15WS * SYNC	TO	15WS.NNV, 15WS.SNV;
NV.ASSIGN	16ES * SYNC	TO	16ES.NNV, 16ES.SNV;
NV.ASSIGN	16WS * SYNC	TO	16WS.NNV, 16WS.SNV;
NV.ASSIGN	17ES * SYNC	TO	17ES.NNV, 17ES.SNV;

NV.ASSIGN	17WS * SYNC	TO	17WS.NNV, 17WS.SNV;
NV.ASSIGN	20ES * SYNC	TO	20ES.NNV, 20ES.SNV;
NV.ASSIGN	20WS * SYNC	TO	20WS.NNV, 20WS.SNV;
NV.ASSIGN	21ES * SYNC	TO	21ES.NNV, 21ES.SNV;
NV.ASSIGN	21WS * SYNC	TO	21WS.NNV, 21WS.SNV;

/* **SIGNAL CONTROL** */

NV.ASSIGN	311H * SYNC	TO	311H.NNV, 311H.SNV;
NV.ASSIGN	312H * SYNC	TO	312H.NNV, 312H.SNV;
NV.ASSIGN	331H * SYNC	TO	331H.NNV, 331H.SNV;
NV.ASSIGN	332H * SYNC	TO	332H.NNV, 332H.SNV;
NV.ASSIGN	333H * SYNC	TO	333H.NNV, 333H.SNV;
NV.ASSIGN	334H * SYNC	TO	334H.NNV, 334H.SNV;
NV.ASSIGN	340H * SYNC	TO	340H.NNV, 340H.SNV;
NV.ASSIGN	361H * SYNC	TO	361H.NNV, 361H.SNV;
NV.ASSIGN	361L * SYNC	TO	361L.NNV, 361L.SNV;
NV.ASSIGN	362H * SYNC	TO	362H.NNV, 362H.SNV;
NV.ASSIGN	362L * SYNC	TO	362L.NNV, 362L.SNV;
NV.ASSIGN	431H * SYNC	TO	431H.NNV, 431H.SNV;
NV.ASSIGN	432H * SYNC	TO	432H.NNV, 432H.SNV;
NV.ASSIGN	460H * SYNC	TO	460H.NNV, 460H.SNV;
NV.ASSIGN	460L * SYNC	TO	460L.NNV, 460L.SNV;

NV.ASSIGN	501H * SYNC	TO	501H.NNV, 501H.SNV;
NV.ASSIGN	512H * SYNC	TO	512H.NNV, 512H.SNV;
NV.ASSIGN	531H * SYNC	TO	531H.NNV, 531H.SNV;
NV.ASSIGN	532H * SYNC	TO	532H.NNV, 532H.SNV;
NV.ASSIGN	533H * SYNC	TO	533H.NNV, 533H.SNV;
NV.ASSIGN	534H * SYNC	TO	534H.NNV, 534H.SNV;
NV.ASSIGN	540H * SYNC	TO	540H.NNV, 540H.SNV;
NV.ASSIGN	611H * SYNC	TO	611H.NNV, 611H.SNV;
NV.ASSIGN	612H * SYNC	TO	612H.NNV, 612H.SNV;

END LOGIC

END PROGRAM

Example 8

Appendix C

MICROLOK_II PROGRAM;

INTERFACE

LOCAL

BOARD: VO_SLOT_J15

ADJUSTABLE ENABLE: 1
TYPE: OUT16

OUTPUT:

```
/*  Only SYNC.OUT, OUT.02, OUT.H.03, and OUT.L.04 simulate true outputs. The remaining
    bits are only used to give an indication on the output board for testing purposes.  */
```

```
SYNC.OUT,                OUT.02,                  OUT.H.03,                OUT.L.04,
SPARE,                   SPARE,                   OUT.SYNC.WAIT.07,        OUT.SYNC.08,
OUT.HEALTH.WAIT.DELAY.09, OUT.HEALTH.WAIT.10,     SPARE,                   OUT.OUT.L.04.12,
OUT.IN.L.04.13,          OUT.OUT.RESET.14,        OUT.IN.RESET.15,         OUT.COMALT.16;
```

BOARD: VI_SLOT_J13

ADJUSTABLE ENABLE: 1
TYPE: IN16

INPUT:

```
/*  In the Normal unit, NORMAL (bit 16) is energized from a constant source. It must be high
    in the Normal unit and low in the Standby unit. SYNC.IN is from the SYNC.OUT of the
    other unit.  */
```

```
SYNC.IN,                 IN.02,                   IN.03,                   IN.04,
VCOR,                    SPARE,                   SPARE,                   SPARE,
SPARE,                   SPARE,                   SPARE,                   SPARE,
SPARE,                   SPARE,                   SPARE,                   NORMAL;
```

COMM

```
/*  Two COM ports are used so that the same software can be used in both units.  */
```

LINK: HOT_MASTER

ADJUSTABLE	ENABLE:	1
	PROTOCOL:	MICROLOK.MASTER
ADJUSTABLE	POINT.POINT:	1;
ADJUSTABLE	PORT:	1;
ADJUSTABLE	BAUD:	19200;
ADJUSTABLE	STOPBITS:	1;
ADJUSTABLE	PARITY:	NONE;
ADJUSTABLE	KEY.ON.DELAY:	12;
ADJUSTABLE	KEY.OFF.DELAY:	12;
ADJUSTABLE	STALE.DATA.TIMEOUT:	3:SEC;
ADJUSTABLE	POLLING.INTERVAL:	50:MSEC;
ADJUSTABLE	MASTER.TIMEOUT:	100:MSEC;

ADDRESS: 1

ADJUSTABLE ENABLE: 1

OUTPUT:

/* SL.OUT.02, SL.OUT.03, and SL.OUT.04 represent all output bits. If the Microlok had two 16 bit Vital Output Boards (and all bits were used) then there would be 32 bits listed. SL.OUT.HEALTH, SL.OUT.RESET, and SL.OUT.SYNC are the only extra bits required for Hot Standby operation. */

SL.OUT.02, SL.OUT.H.03, SL.OUT.L.04,

SL.OUT.HEALTH, SL.OUT.RESET, SL.OUT.SYNC;

LINK: HOT_SLAVE

ADJUSTABLE	ENABLE:	1
	PROTOCOL:	MICROLOK.SLAVE
ADJUSTABLE	POINT.POINT:	0;
ADJUSTABLE	PORT:	2;
ADJUSTABLE	BAUD:	19200;
ADJUSTABLE	STOPBITS:	1;
ADJUSTABLE	PARITY:	NONE;
ADJUSTABLE	KEY.ON.DELAY:	12;
ADJUSTABLE	KEY.OFF.DELAY:	12;
ADJUSTABLE	STALE.DATA.TIMEOUT:	3:SEC;

ADDRESS: 1

ADJUSTABLE ENABLE: 1

INPUT:

/* SL.IN.02, SL.IN.03, and SL.IN.04 represent all input bits from the other unit. If the Microlok had two 16 bit Vital Output Boards (and all bits were used) then there would be 32 bits listed. SL.IN.HEALTH, SL.IN.RESET, and SL.IN.SYNC are the only extra bits required for Hot Standby operation. */

SL.IN.02, SL.IN.H.03, SL.IN.L.04,

SL.IN.HEALTH, SL.IN.RESET, SL.IN.SYNC;

BOOLEAN BITS

SYS.RESET, GROUP.01.RESET, GROUP.02.RESET, GROUP.03.V.RESET

OUT.RESET.02, OUT.H.RESET.03, OUT.L.RESET.04,

SYNC, SYNC.WAIT, STAND.ALONE.SYNC.DELAY, STAND.ALONE.SYNC,

OUT.02.SYNC, OUT.H.03.SYNC, OUT.L.04.SYNC,

DEFAULT.NORMAL.SYNC,

QUICK.HEALTH.START, QUICK.HEALTH,

SL.IN.HEALTH.0, SL.IN.HEALTH.1, IN.HEALTH,

HEALTH.WAIT.DELAY, HEALTH.WAIT,

COMALT,

SL.IN.H.03.D, SL.IN.L.04.D;

TIMER BITS

/* SL.OUT.RESET is sent from the Normal unit to the Standby unit when the Normal unit determines there is a disagreement in bit states. It is delayed to allow the Standby unit time to synchronize. The exact setting for this bit is based on the needs of each application. It should be as short as possible without effecting reliability. */

SL.OUT.RESET:	SET = 3:SEC	CLEAR = 0:SEC;

/* SYS.RESET is an internal bit that RESETS the Standby unit if it is out of synchronization with the online Normal unit. It is slightly delayed to insure that the Standby unit does not falsely reset. The exact setting for this bit is based on the needs of each application. It should be as short as possible without effecting reliability. */

SYS.RESET:	SET = 3:SEC	CLEAR = 0:SEC;

/* The GROUP.RESET bits represent groups of individual bit resets. They are slightly delayed to insure that the Standby unit does not reset falsely. GROUP.03.V.RESET contains bits that are "more vital" such as Switch Locking or Route Locking, therefore they are given a shorter reset time. The exact setting for these bits is based on the needs of each application. They should be as short as possible without effecting reliability. */

GROUP.01.RESET:	SET = 3:SEC	CLEAR = 0:SEC;
GROUP.02.RESET:	SET = 3:SEC	CLEAR = 0:SEC;
GROUP.03.V.RESET:	SET = 1:SEC	CLEAR = 0:SEC;

/* HEALTH.WAIT.DELAY is an internal bit that allows the Standby unit to maintain its outputs while the Normal unit is brought online. It fills in the gap between the time the Normal's VCOR picks and communication between the pair is established. It is set for 20 seconds because it takes approximately 15 seconds for a unit to establish serial communication after the VCOR is picked. */

HEALTH.WAIT.DELAY:	SET = 0:SEC	CLEAR = 20:SEC;

/* HEALTH.WAIT shortens the effect of HEALTH.WAIT.DELAY to 1 second after serial communication is established. HEALTH.WAIT is used in all output bit assign statements. */

HEALTH.WAIT:	SET = 0:SEC	CLEAR = 1:SEC;

/* SL.OUT.HEALTH is the toggling health bit sent to the other unit. */

ADJUSTABLE	SL.OUT.HEALTH:	SET = 1:SEC	CLEAR = 1:SEC;

/* HEALTH.WAIT shortens the effect of HEALTH.WAIT.DELAY to 1 second after serial communication is established. HEALTH.WAIT is used in all output bit assign statements. */

ADJUSTABLE	IN.HEALTH:	SET = 4:SEC	CLEAR = 1:SEC;
ADJUSTABLE	SL.IN.HEALTH.0:	SET = 0:SEC	CLEAR = 2:SEC;
ADJUSTABLE	SL.IN.HEALTH.1:	SET = 0:SEC	CLEAR = 2:SEC;

/* HEALTH.WAIT shortens the effect of HEALTH.WAIT.DELAY to 1 second after serial communication is established. HEALTH.WAIT is used in all output bit assign statements. */

ADJUSTABLE	QUICK.HEALTH.ST	SET = 0:SEC	CLEAR = 20:SEC;
ADJUSTABLE	QUICK.HEALTH:	SET = 0:SEC	CLEAR = 1:SEC;

/* STAND.ALONE.SYNC.DELAY is a slow set bit that allows the unit to stabilize before VCOR is referenced for SYNC. */

STAND.ALONE.SYNC.DELAY:	SET = 1:SEC	CLEAR = 0:SEC;

/* SYNC.WAIT is a slow set internal bit that allows serial communication to stabilize after the unit is powered up before synchronization is verified. It should always be set for 5 seconds or longer. */

SYNC.WAIT:	SET = 5:SEC	CLEAR = 0:SEC;

```
/*  SL.IN.H.03.D and SL.IN.L.04.D are test bits used to simulate the delay in serial
    communications between the units.                                              */
```

SL.IN.H.03.D:	SET = 1:SEC	CLEAR = 0:SEC;
SL.IN.L.04.D:	SET = 1:SEC	CLEAR = 0:SEC;

CONSTANTS BOOLEAN

ONE	=	1;
ZERO	=	0;

CONFIGURATION

SYSTEM

ADJUSTABLE	DEBUG_PORT_ADDRESS:	1;
ADJUSTABLE	DEBUG_PORT_BAUDRATE:	9600;
ADJUSTABLE	LOGIC_TIMEOUT:	2:SEC;
ADJUSTABLE	DELAY_RESET:	3:SEC;

LOGIC BEGIN

/* UTILITY BITS */

ASSIGN	ONE	TO	CPS.ENABLE;
ASSIGN	ONE	TO	STAND.ALONE.SYNC.DELAY;
ASSIGN	ONE	TO	SL.OUT.HEALTH;
ASSIGN	SYS.RESET	TO	RESET;

/* RESET BITS */

```
/*  SYS.RESET is a slow set bit that only functions in the Standby unit.
    The bits in the assign statement function as follows:
    • ~NORMAL insures that only the Standby unit can be RESET.
    • VCOR insures that the unit will only RESET if the Normal unit is online.
    • SL.IN.RESET comes from the Normal unit and forces the Standby unit to RESET.
    • ~SL.IN.HEALTH insures the Standby unit will RESET itself if serial communication is
      lost between the units.
    • ~HEALTH.WAIT insures the Standby unit will not RESET itself before serial
      communication is established when the Normal unit is coming online.
    •                                                                              */
```

- SL.IN.SYNC insures that the Standby unit will only RESET itself if the Normal unit is in sync.
- ~SYNC insures that the Standby unit will RESET itself if both units are powered up simultaneously and do not achieve synchronization. This permits the Normal unit to take control.
- SYNC.WAIT delays the RESET until the Standby unit has an opportunity to verify synchronization with the Normal unit.
- GROUP.01.RESET, GROUP.02.RESET, and GROUP.03.V.RESET are groups of individual reset bits.

```
ASSIGN  ~NORMAL * VCOR * SYNC * SYNC.IN *
        (SL.IN.RESET + (~IN.HEALTH * SL.IN.SYNC *
        ~QUICK.HEALTH) *                          TO  SYS.RESET;
        (GROUP.01.RESET + GROUP.02.RESET +
        GROUP.03.V.RESET))
```

```
/*  SL.OUT.RESET is a slow set bit that is sent from the Normal unit to the Standby unit when
    any output bit is out of sync. It is primarily controlled by the GROUP.RESET bits, however,
    the SYNC bit is also required so that the Normal unit cannot reset the Standby unit if the
    Normal is being powered up and cannot achieve synchronization with the Standby.          */
```

```
ASSIGN  (NORMAL * SYNC) * ~IN.HEALTH *
        (GROUP.01.RESET + GROUP.02.RESET +        TO  SL.OUT.RESET;
        GROUP.03.V.RESET)
```

```
/*  GROUP.01.RESET, GROUP.02.RESET, and GROUP.03.V.RESET are groups of
    individual reset bits (though only one RESET bit is assigned to each for testing). The
    individual reset bits are grouped together to simplify the SYS.RESET equation and to allow
    for a longer time delay for non-synchronous situations which may be caused by serial
    communication delays. Three groups are used for testing but the maximum number of
    groups is unlimited. Multiple groups should be used to limit the number of bits so that
    continuous changes of bit states will not by misinterpreted as a non-synchronous condition.
    GROUP.03.V.RESET represents groups of bits that are "more vital" such as Switch Locks
    and Route Locks. This group is given the absolutely shortest time delay possible while still
    maintaining reliability.                                                                 */
```

```
ASSIGN  OUT.RESET.02      TO  GROUP.01.RESET;
ASSIGN  OUT.H.RESET.03    TO  GROUP.02.RESET;
ASSIGN  OUT.L.RESET.04    TO  GROUP.03.V.RESET;
```

/* SYNCHRONIZATION BITS */

```
/*  SYNC suppresses all outputs of the unit being brought online until they are verified to be
    synchronous with the unit currently in control or the other unit's VCOR is down. Once it is
    set it is stuck high until the unit is powered down. SL.OUT.SYNC is sent out to the other
    unit. It is utilized by the Standby unit in the SYS.RESET assign statement.              */
```

```
ASSIGN  SYNC + STAND.ALONE.SYNC +
        (OUT.02.SYNC * OUT.H.03.SYNC *
        OUT.L.04.SYNC)                          TO  SYNC,
                                                    SL.OUT.SYNC,
                                                    SYNC.OUT,
                                                    LED.1;
```

```
/*  SYNC.WAIT is a slow set bit that suppresses verification of bits in the unit being brought
    online until it is powered up and both the unit and the serial communication link are stable.
    Once it is set it is stuck high until the unit is powered down.                          */
```

```
ASSIGN  SYNC.WAIT + (VCOR * SL.IN.HEALTH)       TO  SYNC.WAIT;
```

```
/*  STAND.ALONE.SYNC is an internal bit that will set the SYNC bit one second after the
    unit is powered up if the other unit's VCOR is down.                                     */
```

```
ASSIGN  ~VCOR * STAND.ALONE.SYNC.DELAY *
        ~SYNC.IN * ~SL.IN.SYNC +
        DEFAULT.NORMAL.SYNC                     TO  STAND.ALONE.SYNC;
```

```
/*  DEFAULT.NORMAL.SYNC is a slow set bit that lets the Normal unit achieve SYNC if
    both units are powered on simultaneously and there is a disagreement between the units.  */
```

```
ASSIGN  VCOR * STAND.ALONE.SYNC.DELAY *
        ~SYNC.IN * ~SL.IN.SYNC * NORMAL *
        IN.HEALTH                               TO  DEFAULT.NORMAL.SYNC;
```

/* HEALTH BITS */

```
/*  HEALTH.WAIT and HEALTH.WAIT.DELAY allow the Standby unit to maintain its
    outputs between the time that the Normal unit's VCOR picks and communication is
    established between the units. HEALTH.WAIT.DELAY is a slow clear bit that sets when
    the VCOR picks in the unit coming online. HEALTH.WAIT.DELAY sets HEALTH.WAIT
    which remains high until either HEALTH.WAIT.DELAY expires or serial communication
    is established. This is necessary to insure that as soon as serial communication is established
    the HEALTH bit is not able to override any logic.                                        */
```

```
ASSIGN  VCOR * ~SL.IN.HEALTH *
        ~HEALTH.WAIT.DELAY                      TO  HEALTH.WAIT.DELAY;

ASSIGN  HEALTH.WAIT.DELAY * ~SL.IN.HEALTH       TO  HEALTH.WAIT;
```

```
/*  SL.OUT.HEALTH is a toggling bit that is sent to the other unit.                          */
```

```
ASSIGN  ~SL.OUT.HEALTH                          TO  SL.OUT.HEALTH;
```

```
/*  QUICK.HEALTH.START and QUICK.HEALTH work together to prevent the Standby unit
    from falsely resetting while it is coming online.                                        */
```

ASSIGN	SL.IN.HEALTH * ~IN.HEALTH * ~QUICK.HEALTH	TO	QUICK.HEALTH.START;
ASSIGN	QUICK.HEALTH.START * ~IN.HEALTH	TO	QUICK.HEALTH;

/* SL.IN.HEALTH is received as a toggling bit from the other unit. SL.IN.HEALTH.0, SL.IN.HEALTH.1, and IN.HEALTH work together to verify the stability of the communication link. */

ASSIGN	SL.IN.HEALTH	TO	SL.IN.HEALTH.1;
ASSIGN	~SL.IN.HEALTH	TO	SL.IN.HEALTH.0;
ASSIGN	SL.IN.HEALTH.0 * SL.IN.HEALTH.1	TO	IN.HEALTH;

/* **OUTPUT BITS** */

There are three types of output bits:

1. Unrestricted, represented by OUT.02.

   These bits require no bit specific serial communication between the units in order to produce an output; therefore they are the fastest and should always be utilized whenever possible. They should never be used for signal lighting or any type of locking.

2. Half Restricted, represented by OUT.03.

   These bits are unrestricted in the Normal unit, but restricted in the Standby. The Standby unit cannot produce the output until it receives verification (via serial communication) that the Normal unit has also satisfied the assign statement. This type of bit is specifically designed for signal lighting. If the bits are out of sync, it can only be that the Normal unit has the aspect lit and the Standby does not. In this event the Standby unit is reset, and the signal aspect will not change.

/* 3. Restricted, represented by OUT.04. */

   These bits are restricted in both the Normal and the Standby units. Neither unit can produce the output until it receives verification (via serial communication) that the other unit has also satisfied the assign statement. This type of bit is the slowest due to the amount of serial communication involved. It was specifically designed for locking. The bit cannot be set (unlocked) until both units satisfy the assign statement and it will be cleared (locked) immediately at any time that the units do not agree.

The three types of bits have the following in common:

- If the other unit's VCOR is down the unit will produce the output whenever the assign statement is satisfied.
- If the other unit's VCOR is up the unit must also receive serial communication.
  - Unrestricted bits require a generic health bit.

- Half-Restricted bits require a bit verification from Normal to Standby.
- Restricted bits require bit verification to and from both units.

- If the other unit is in control, the unit being brought online cannot produce any output until it is in SYNC.
- If both units are online and any bit becomes out of sync for a selected period of time either the Normal unit will reset the Standby or the Standby will reset itself.

```
/*  UNRESTRICTED BITS                                                    */
```

```
/*  The term IN.02 is used for testing purposes. In reality it would be replaced by a logic
    equation. In the term SL.OUT.02, SL stands for Serial Link, and OUT.02 represents the
    resulting bit of the satisfied assign statement. SL.OUT.02 is immediately sent out serially to
    the other unit.                                                      */
```

```
ASSIGN   IN.02                          TO   SL.OUT.02;
```

```
/*  OUT.02.SYNC is primarily satisfied by the SL.OUT.02 bit. If the other unit is online
    (VCOR is picked, or in the process of booting up) it is referenced to insure that the bit is in
    the same state. Once it is set it is stuck high until the unit is powered down. If the other unit
    is offline (VCOR down) this bit is bypassed and the SYNC bit assign statement is satisfied
    with STAND.ALONE.SYNC.                                               */
```

```
ASSIGN   OUT.02.SYNC +
         (((SL.OUT.02 * SL.IN.02) +
         (~SL.OUT.02 * ~SL.IN.02))
          * VCOR * SYNC.WAIT)                TO   OUT.02.SYNC;
```

```
/*  OUT.02 is the bit that sets the output high on the Vital Output Board. It is primarily satisfied
    by the SL.OUT.02 bit. In the Normal unit the only other requirement is that the SYNC bit
    must be set (which it will be unless the Normal is in the process of coming online). The
    Standby unit requires a serial communication HEALTH bit or HEALTH.WAIT.
    HEALTH.WAIT is used keep the Standby unit's outputs set between the time the Normal
    unit's VCOR picks and serial communication is established when the Normal unit is being
    brought online. Both the Normal and Standby units will immediately set the bit if
    SL.OUT.02 is high and the other unit is offline (VCOR down).         */
```

```
ASSIGN   SL.OUT.02 * (SYNC * (NORMAL +
         (~NORMAL * (SL.IN.HEALTH +                TO   OUT.02;
         HEALTH.WAIT))) + ~VCOR)
```

```
/*  OUT.RESET.02 causes the Standby unit to reset if there is a disagreement in the bit state
    between the units.                                                   */
```

```
ASSIGN   (SL.OUT.02 * ~SL.IN.02) + (~SL.OUT.02 * SL.IN.02)   TO   OUT.RESET.02;
```

/* **HALF RESTRICTED BITS** */

/* The logic statement for OUT.H.03 functions the same as OUT.02 above, with one exception. In the statement for OUT.H.03 the generic serial communication HEALTH bit is replaced with the corresponding bit (SL.IN.H.03.D) from the Normal unit. This suppresses the output from the Standby unit until it has been verified that the Normal has also satisfied the assign statement. SL.IN.H.03.D is a slow set bit used for testing to simulate serial communication delays. */

ASSIGN	IN.03	TO	SL.OUT.H.03;
ASSIGN	OUT.H.03.SYNC + (((SL.OUT.H.03 * SL.IN.H.03) + (~SL.OUT.H.03 * ~SL.IN.H.03)) * VCOR * SYNC.WAIT)	TO	OUT.H.03.SYNC;
ASSIGN	SL.OUT.H.03 * (SYNC * (NORMAL + (~NORMAL * (SL.IN.H.03.D + HEALTH.WAIT))) + ~VCOR)	TO	OUT.H.03;
ASSIGN	(SL.OUT.H.03 * ~SL.IN.H.03) + (~SL.OUT.H.03 * SL.IN.H.03)	TO	OUT.H.RESET.03;

/* **RESTRICTED BITS** */

/* The logic statements for OUT.L.04 are the same as OUT.H.03 above, with one exception. In the statement for OUT.L.04 there are no separate variables for Normal or Standby. Both units must have SL.OUT.L.04, be in SYNC, and receive the corresponding bit from the other unit. This suppresses the output from either unit until it has been verified that the other unit has also satisfied the assign statement and immediately drops the output if it loses the verification from the other unit. */

ASSIGN	IN.04	TO	SL.OUT.L.04;
ASSIGN	OUT.L.04.SYNC + (((SL.OUT.L.04 * SL.IN.L.04) + (~SL.OUT.L.04 * ~SL.IN.L.04)) * VCOR * SYNC.WAIT)	TO	OUT.L.04.SYNC;
ASSIGN	SL.OUT.L.04 * (SYNC * (SL.IN.L.04.D + HEALTH.WAIT) + ~VCOR)	TO	OUT.L.04;
ASSIGN	(SL.OUT.L.04 * ~SL.IN.L.04) + (~SL.OUT.L.04 * SL.IN.L.04)	TO	OUT.L.RESET.04;

/* **COMMUNICATION ALERT BIT** */

/* COMALT may be used to alert central control of any problems in the communication between the Normal and Standby units. */

ASSIGN	~SL.IN.HEALTH	TO	COMALT;

/* **FOR TEST PURPOSES ONLY** */

/* **COMMUNICATION DELAY SIMULATORS** */

/* SL.IN.H.03.D and SL.IN.L.04.D are slow set bits that simulate the possible delay in the serial communication between the Normal and Standby units. */

ASSIGN	SL.IN.H.03	TO	SL.IN.H.03.D;
ASSIGN	SL.IN.L.04	TO	SL.IN.L.04.D;

/* **BIT MONITORS** */

/* The following bits produce indications on the Vital Output Board for testing purposes. */

ASSIGN	SYNC.WAIT	TO	OUT.SYNC.WAIT.07;
ASSIGN	SYNC	TO	OUT.SYNC.08;
ASSIGN	HEALTH.WAIT.DELAY	TO	OUT.HEALTH.WAIT.DELAY.09;
ASSIGN	HEALTH.WAIT	TO	OUT.HEALTH.WAIT.10;
ASSIGN	SL.OUT.L.04	TO	OUT.OUT.L.04.12;
ASSIGN	SL.IN.L.04	TO	OUT.IN.L.04.13;
ASSIGN	SL.OUT.RESET	TO	OUT.OUT.RESET.14;
ASSIGN	SL.IN.RESET	TO	OUT.IN.RESET.15;
ASSIGN	COMALT	TO	OUT.COMALT.16;

END LOGIC

END PROGRAM

What is Claimed is:

1. An apparatus for providing hot standby operation, said apparatus comprising:

a normal processor;

a standby processor;

each of said normal and standby processors comprising:

a plurality of vital inputs, at least some of said vital inputs being electrically interconnected with at least some of said vital inputs of the other one of said standby and normal processors,

a plurality of vital outputs,

means for communicating with the other one of said standby and normal processors,

a health routine providing a health status after communication is established with the other one of said standby and normal processors through said means for communicating,

a vital relay including an input controlled by one of said vital outputs and an output to one of said vital inputs of the other one of said standby and normal processors,

a synchronization routine providing a synchronization status through said means for communicating with the other one of said standby and normal processors, and

an application routine outputting said vital outputs when said synchronization status is set and inputting said vital inputs;

said standby processor further comprising a reset routine, which resets said standby processor when said health status of said standby processor is not provided; and

means for outputting from some of said vital outputs of said normal processor and from some of said vital outputs of said standby processor.

2. The apparatus of Claim 1 wherein said normal and standby processors operate in at least one mode selected from the group comprising:

a first mode wherein both of said normal and standby processors output through at least one of said some of said vital outputs of said normal and standby processors, respectively, without restriction;

a second mode wherein said normal processor outputs through at least one of said some of said vital outputs of said normal processor without restriction and said standby processor verifies through said means for communicating of said standby processor that said standby processor agrees with said normal processor before outputting through at least one of said some of said vital outputs of said standby processor and, otherwise, said standby processor being reset; and

a third mode wherein both of said normal and standby processors verify through said means for communicating of said normal and standby processors, respectively, that said normal and standby processors, respectively, agree with said standby and normal processors, respectively, before outputting through at least one of said some of said vital outputs of said normal and standby processors, respectively, and, otherwise, said normal and standby processors being reset.

3. The apparatus of Claim 2 wherein for said first mode the application routines of said normal and standby processors enable said some of said vital outputs of said normal and standby processors, respectively, if the output of said vital relay of said standby and normal processors, respectively, is set and if the health routine of said normal and standby processors, respectively, determines said health status.

4. The apparatus of Claim 2 wherein for said second mode the application routine of said normal processor enables said some of said vital outputs of said normal processor if the output of said vital relay of said standby processor is set and if the health routine of said normal processor determines said health status.

5. The apparatus of Claim 2 wherein for said second mode the application routine of said standby processor enables one of said some of said vital outputs of said standby processor if the output of said vital relay of said normal processor is set, if the health routine of said standby processor determines said health status, and if said standby processor verifies through said means for communicating of said standby processor that said one of said some of said vital outputs of said

standby processor agrees with a corresponding one of said some of said vital outputs of said normal processor.

6. The apparatus of Claim 2 wherein for said third mode the application routines of said normal and standby processors enable one of said some of said vital outputs of said normal and standby processors, respectively, if the output of said vital relay of said standby and normal processors, respectively, is set, and if the health routine of said normal and standby processors, respectively, determines said health status, and if said normal and standby processors verify through said means for communicating of said normal and standby processors, respectively, that said one of said some of said vital outputs of said normal and standby processors, respectively, agrees with said standby and normal processors, respectively.

7. The apparatus of Claim 1 wherein both of said normal and standby processors operate in a mode in which said normal and standby processors output through at least one of said some of said vital outputs of said normal and standby processors, respectively, without restriction.

8. The apparatus of Claim 1 wherein said normal and standby processors operate in modes wherein said normal processor outputs through at least one of said some of said vital outputs of said normal processor without restriction and said standby processor verifies through said means for communicating of said standby processor that said standby processor agrees with said normal processor before outputting through at least one of said some of said vital outputs of said standby processor and, otherwise, said standby processor being reset.

9. The apparatus of Claim 8 wherein the application routine of said standby processor enables one of said some of said vital outputs of said standby processor if the health routine of said standby processor determines said health status and if said standby processor verifies through said means for communicating of said standby processor that said one of said some of said vital outputs of said standby processor agrees with a corresponding one of said some of said vital outputs of said normal processor.

10. The apparatus of Claim 8 wherein said some of said vital outputs of said normal and standby processors include signal lighting outputs.

11. The apparatus of Claim 1 wherein said normal and standby processors operate in mode wherein both of said normal and standby processors verify through said means for communicating of said normal and standby processors, respectively, that said normal and standby processors, respectively, agree with said standby and normal processors, respectively, before outputting through at least one of said some of said vital outputs of said normal and standby processors, respectively, and, otherwise, said normal and standby processors are reset.

12. The apparatus of Claim 11 wherein said some of said vital outputs of said normal and standby processors include lock outputs.

13. The apparatus of Claim 12 wherein said lock outputs include a first lock output and a second lock output; wherein the first lock output of said normal processor is in agreement with the first lock output of said standby processor; wherein both of said first lock outputs are set to an unlocked state; wherein the second lock output of said normal processor is not in agreement with second lock output of said standby processor; and wherein both of said second lock outputs are set to a locked state.

14. The apparatus of Claim 12 wherein said lock outputs include a first lock output and a second lock output; wherein the first lock output of said normal processor is in agreement with the first lock output of said standby processor; wherein both of said first lock outputs are set to a locked state; wherein the second lock output of said normal processor is not in agreement with second lock output of said standby processor; and wherein both of said second lock outputs are set to a locked state.

15. The apparatus of Claim 2 wherein both of said normal and standby processors are operating and capable of outputting through said means for outputting to a single output device.

16. The apparatus of Claim 15 wherein said means for outputting includes a first diode having an anode and a cathode and a second diode having an anode and a cathode; wherein said cathodes are adapted for electrical connection to said single output device; wherein the anode of said first diode is electrically connected to one of said some of said vital outputs of said normal processor; and wherein the anode of said second diode is electrically connected to a corresponding one of said some of said vital outputs of said standby processor.

17. The apparatus of Claim 15 wherein said means for outputting includes a first diode array having an input and an output and a second diode array having an input and an output; wherein said outputs of said first and second diode arrays are adapted for electrical connection to said single output device; wherein the input of said first diode array is electrically connected to one of said some of said vital outputs of said normal processor; and wherein the input of said second diode array is electrically connected to a corresponding one of said some of said vital outputs of said standby processor.

18. The apparatus of Claim 17 wherein each of said first and second diode arrays includes a first pair of series-connected diodes and a second pair of series-connected diodes, said first pair of series-connected diodes being electrically connected in parallel with said second pair of series-connected diodes, said first and second pairs having a pair of anodes as the input of the corresponding one of said first and second diode arrays, said first and second pairs having a pair of cathodes as the output of the corresponding one of said first and second diode arrays.

19. The apparatus of Claim 1 wherein said means for outputting includes a vital OR circuit having a first input from one of said some of said vital outputs of said normal processor, a second input from one of said some of said vital outputs of said standby processor, and an output adapted to output to a single output device.

20. The apparatus of Claim 1 wherein said means for communicating includes at least one communication port adapted for communication with the other one of said standby and normal processors.

21. The apparatus of Claim 20 wherein said at least one communication port is at least one serial communication port.

22. The apparatus of Claim 21 wherein said at least one serial communication port includes an output serial communication port for outputting serial data from one of said standby and normal processors to the other of said standby and normal processors, and further includes an input serial communication port for inputting serial data from the other one of said standby and normal processors to said one of said standby and normal processors.

23. The apparatus of Claim 1 wherein said normal and standby processors form an interlocking control system.

24. The apparatus of Claim 1 wherein said health routine of said normal and standby processors periodically exchanges health information with said health routine of said standby and normal processors, respectively, in order to provide said health status when said one of said vital inputs of the other one of said standby and normal processors is set and said health information is periodically received.

25. The apparatus of Claim 24 wherein said normal and standby processors operate in a mode wherein both of said normal and standby processors verify through said means for communicating of said normal and standby processors, respectively, that said normal and standby processors, respectively, agree with said standby and normal processors, respectively, before outputting through at least one of said some of said vital outputs of said normal and standby processors, respectively, and, otherwise, said normal and standby processors being reset; wherein the application routines of said normal and standby processors enable one of said some of said vital outputs of said normal and standby processors, respectively, if the output of said vital relay of said standby and normal processors, respectively, is set, and if the health routine of said normal and standby processors, respectively, determines said health status, and if said normal and standby processors verify through said means for communicating of said normal and standby processors, respectively, that said one of said some of said vital outputs of said normal and standby processors, respectively, agree with said standby and normal processors, respectively.

26. The apparatus of Claim 24 wherein the health routine of said normal processor outputs a reset command to said standby processor whenever the input of said vital relay controlled by said one of said vital outputs of said normal processor is set and the synchronization routine of said normal processor loses said synchronization status; and wherein said reset routine of said standby processor employs said reset command to reset said standby processor.

27. The apparatus of Claim 26 wherein said reset routine of said standby processor initially ignores said reset command when said synchronization status is not set.

28. The apparatus of Claim 1 wherein said application routine outputs said vital outputs: (a) when said one of said vital inputs of the other one of said standby and normal processors is not set; and (b) when said one of said vital inputs of the other one of said standby and normal processors is set and said synchronization status is set.

29. The apparatus of Claim 1 wherein said one of said vital inputs of both of said normal and standby processors is set; wherein said standby processor verifies through said means for communicating of said standby processor that said standby processor disagrees with said normal processor before outputting through at least one of said some of said vital outputs of said standby processor; and wherein said reset routine of said standby processor resets said standby processor.

30. The apparatus of Claim 1 wherein said one of said vital inputs of both of said normal and standby processors is set; wherein said normal processor verifies through said means for communicating of said normal processor that said standby processor disagrees with said normal processor before outputting through at least one of said some of said vital outputs of said normal processor; and wherein said reset routine of said normal processor outputs a reset command through said means for communicating of said normal processor to reset said standby processor.

31. The apparatus of Claim 1 wherein said synchronization routine of said normal and standby processors sets said synchronization status when said one of said vital inputs of the other one of said standby and normal processors, respectively, is set, and when said normal and standby processors, respectively, verifies through said means for communicating that said one of said some of said vital outputs of said normal and standby processors, respectively, agrees with the corresponding one of said some of said vital outputs of said standby and normal processors, respectively.

32. The apparatus of Claim 1 wherein when said synchronization status of one of said standby and normal processors is not set, when said one of said vital inputs of the other one of said standby and normal processors is set, and when said one of said normal and standby processors, respectively, verifies through said means for communicating of said normal and standby processors, respectively, that

said standby processor disagrees with said normal processor, said at least one of said some of said vital outputs is disabled.

33. A hot standby method comprising:

employing a normal processor;

employing a standby processor;

with each of said normal and standby processors:

employing a plurality of vital inputs,

electrically interconnecting at least some of said vital inputs with at least some of said vital inputs of the other one of said standby and normal processors,

employing a plurality of vital outputs,

communicating with the other one of said standby and normal processors,

providing a health status after communication is established with the other one of said standby and normal processors,

employing a vital relay including an input controlled by one of said vital outputs and an output to one of said vital inputs of the other one of said standby and normal processors,

providing a synchronization status associated with said communicating with the other one of said standby and normal processors, and

employing an application routine for outputting said vital outputs when said synchronization status is set and inputting said vital inputs;

employing with said standby processor a reset routine, which resets said standby processor when said health status of said standby processor is not provided; and

outputting from some of said vital outputs of said normal processor and from some of said vital outputs of said standby processor.

34. The method of Claim 33 further comprising

operating said normal and standby processors in a mode wherein both of said normal and standby processors output through at least one of said some of said vital outputs of said normal and standby processors, respectively, without restriction; and

enabling said some of said vital outputs of said normal and standby processors, respectively, if the output of said vital relay of said standby and normal processors, respectively, is set and if said normal and standby processors, respectively, determine said health status.

35. The method of Claim 33 further comprising

operating said normal processor in a mode to output through at least one of said some of said vital outputs of said normal processor without restriction; and

operating said standby processor in a mode to verify through said communicating that said standby processor agrees with said normal processor before outputting through at least one of said some of said vital outputs of said standby processor and, otherwise, resetting said standby processor.

36. The method of Claim 35 further comprising

enabling said some of said vital outputs of said normal processor if the output of said vital relay of said standby processor is set and if said normal processor determines said health status.

37. The method of Claim 35 further comprising

enabling one of said some of said vital outputs of said standby processor if the output of said vital relay of said normal processor is set, if said standby processor determines said health status, and if said standby processor verifies through said communicating that said one of said some of said vital outputs of said standby processor agrees with a corresponding one of said some of said vital outputs of said normal processor.

38. The method of Claim 33 further comprising

operating said normal and standby processors in a mode wherein both of said normal and standby processors verify through said communicating that said normal and standby processors, respectively, agree with said standby and normal processors, respectively, before outputting through at least one of said some of said vital outputs of said normal and standby processors, respectively, and, otherwise, resetting said normal and standby processors.

39. The method of Claim 38 further comprising

enabling one of said some of said vital outputs of said normal and standby processors, respectively, if the output of said vital relay of said standby and normal processors, respectively, is set, and if said normal and standby processors, respectively, determine said health status, and if said normal and standby processors verify through said communicating that said one of said some of said vital outputs of said normal and standby processors, respectively, agrees with said standby and normal processors, respectively.

40. The method of Claim 33 further comprising

operating said normal and standby processors in a mode wherein both of said normal and standby processors verify through said communicating that said normal and standby processors, respectively, agree with said standby and normal processors, respectively, before outputting through at least one of said some of said vital outputs of said normal and standby processors, respectively, and, otherwise, resetting said normal and standby processors.

41. The method of Claim 33 further comprising

employing a vital OR circuit having a first input from one of said some of said vital outputs of said normal processor, a second input from one of said some of said vital outputs of said standby processor, and an output adapted to output to a single output device.

42. The method of Claim 33 further comprising

forming an interlocking control system with said normal and standby processors.

43. The method of Claim 33 further comprising

periodically exchanging health information between said standby and normal processors, respectively, in order to provide said health status when said one of said vital inputs of the other one of said standby and normal processors is set and said health information is periodically received.

44. The method of Claim 43 further comprising

ignoring said reset command when said synchronization status is not set.

45. The method of Claim 33 further comprising

outputting said vital outputs: (a) when said one of said vital inputs of the other one of said standby and normal processors is not set; and (b) when said one of said vital inputs of the other one of said standby and normal processors is set and said synchronization status is set.

46. The method of Claim 33 further comprising

setting said one of said vital inputs of both of said normal and standby processors; and

verifying through said communicating that said standby processor disagrees with said normal processor before outputting through at least one of said some of said vital outputs of said standby processor, and responsively resetting said standby processor.

47. A method for providing normal and standby processors, said method comprising:

employing a normal processor;

employing a standby processor;

with each of said normal and standby processors:

employing a plurality of vital inputs,

electrically interconnecting at least some of said vital inputs with at least some of said vital inputs of the other one of said standby and normal processors,

employing a plurality of vital outputs,

communicating with the other one of said standby and normal processors,

providing a health status after communication is established with the other one of said standby and normal processors,

employing a vital relay including an input controlled by one of said vital outputs and an output to one of said vital inputs of the other one of said standby and normal processors,

providing a synchronization status associated with said communicating with the other one of said standby and normal processors, and

employing an application routine for outputting said vital outputs when said synchronization status is set and inputting said vital inputs;

employing with said standby processor a reset routine, which resets said standby processor when said health status of said standby processor is not provided;

outputting from some of said vital outputs of said normal processor and from some of said vital outputs of said standby processor; and

disabling said some of said vital outputs of said standby processor if the output of said vital relay of said normal processor is set.

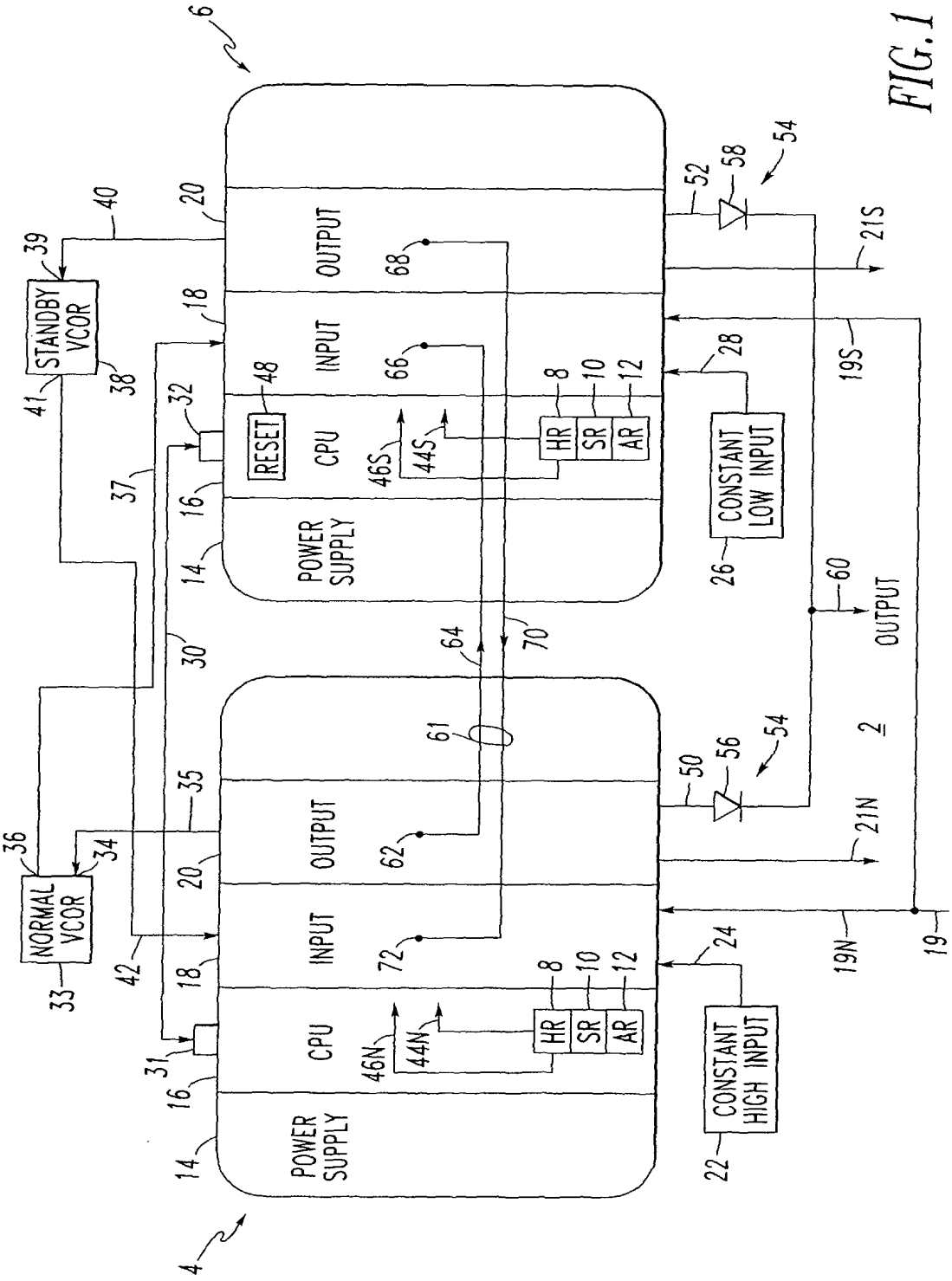

6
4
STANDBY VCOR
NORMAL VCOR
POWER SUPPLY
CPU
INPUT
OUTPUT
RESET
HR
SR
AR
CONSTANT HIGH INPUT
CONSTANT LOW INPUT
OUTPUT
2
21S
21N
19S
19N
19
46S
44S
46N
44N

FIG. 1

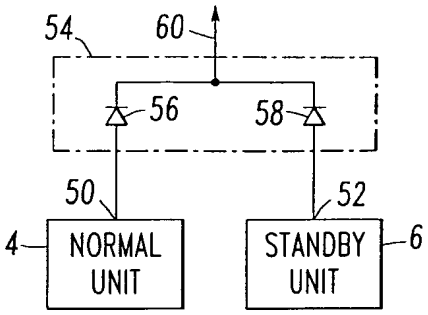


FIG.2

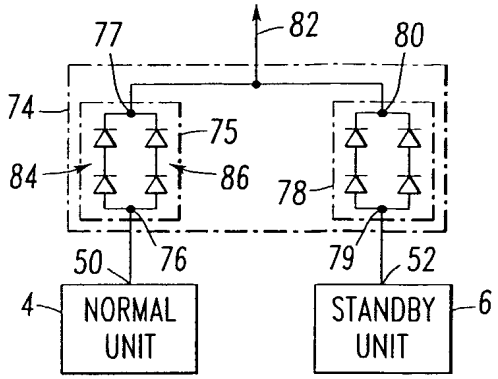


FIG.3

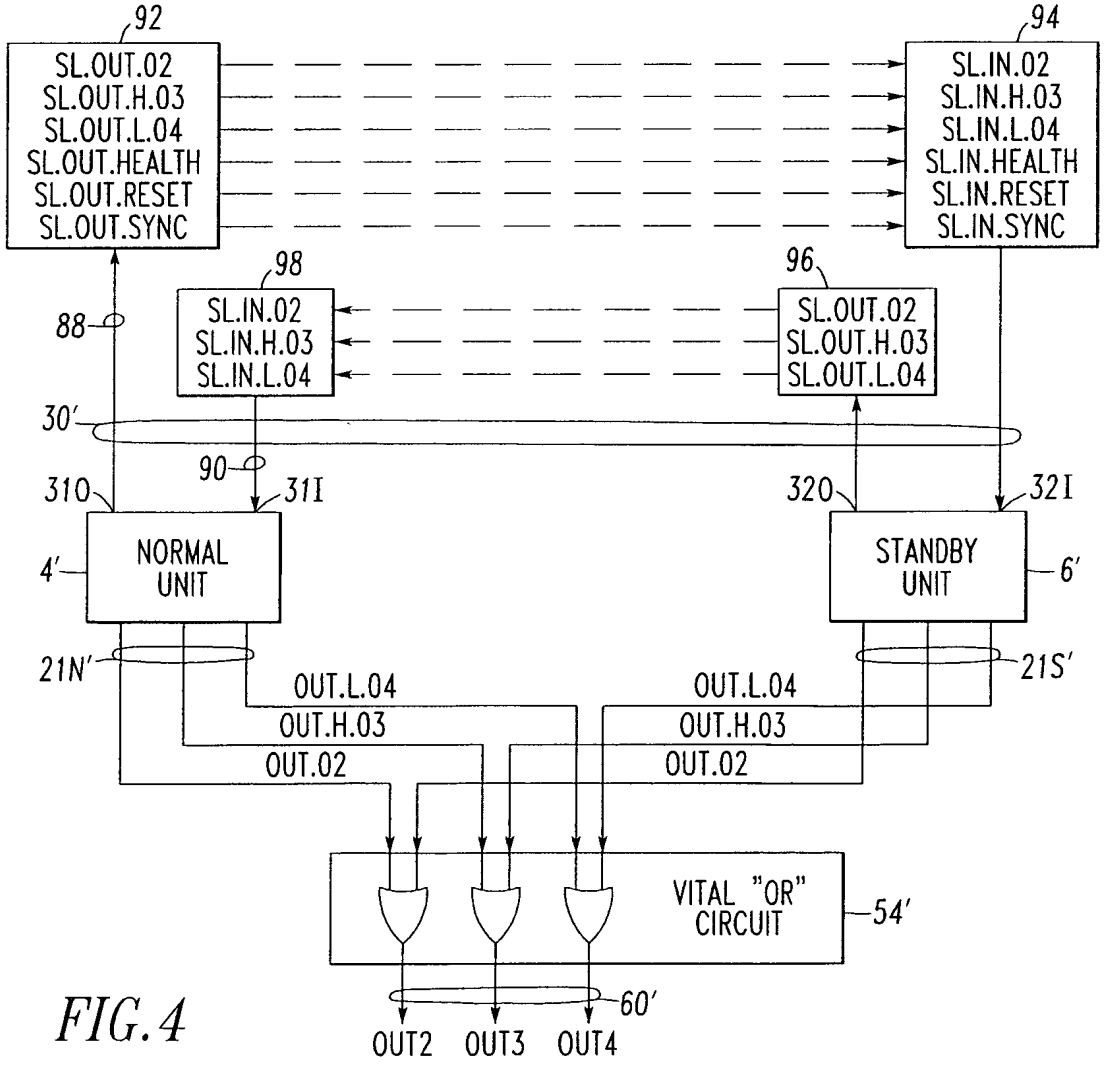


FIG.4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/28149

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 11/00
US CL : 714/13
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 714/13, 10, 11, 12

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A, P	US 6,449,733 B1 (BARTLETT et al) 10 September 2002 (10.09.2002), entire document.	1-47
A	US 6,023,772 A (FLEMING) 8 February 2000 (08.02.2000), entire document.	1-47
A	US 6,202,170 B1 (BUSSCHBACH et al) 13 March 2001 (13.03.2001), entire document.	1-47
A, P	US 6,496,940 B1 (HORST et al) 17 December 2002 (17.12.2002), entire document.	1-47

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 November 2003 (12.11.2003)	09 DEC 2003
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (703)305-3230	Authorized officer Robert Beausoliel Telephone No. 703-305-3900

Form PCT/ISA/210 (second sheet) (July 1998)